Inference of SIG

Song-Chun Zhu, Sinisa Todorovic, and Ales Leonardis

At CVPR, Providence, Rhode Island
June 16, 2012

Typical parsing algorithms in the NLP literature

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, 1980s
   (Baker, Lori, Young)

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

Dynamic Programming (DP)

- Definition: Solve an optimization problem by partitioning it into (simpler) subproblems, and re-use solutions of the subproblems (memoization), rather than re-computing them.

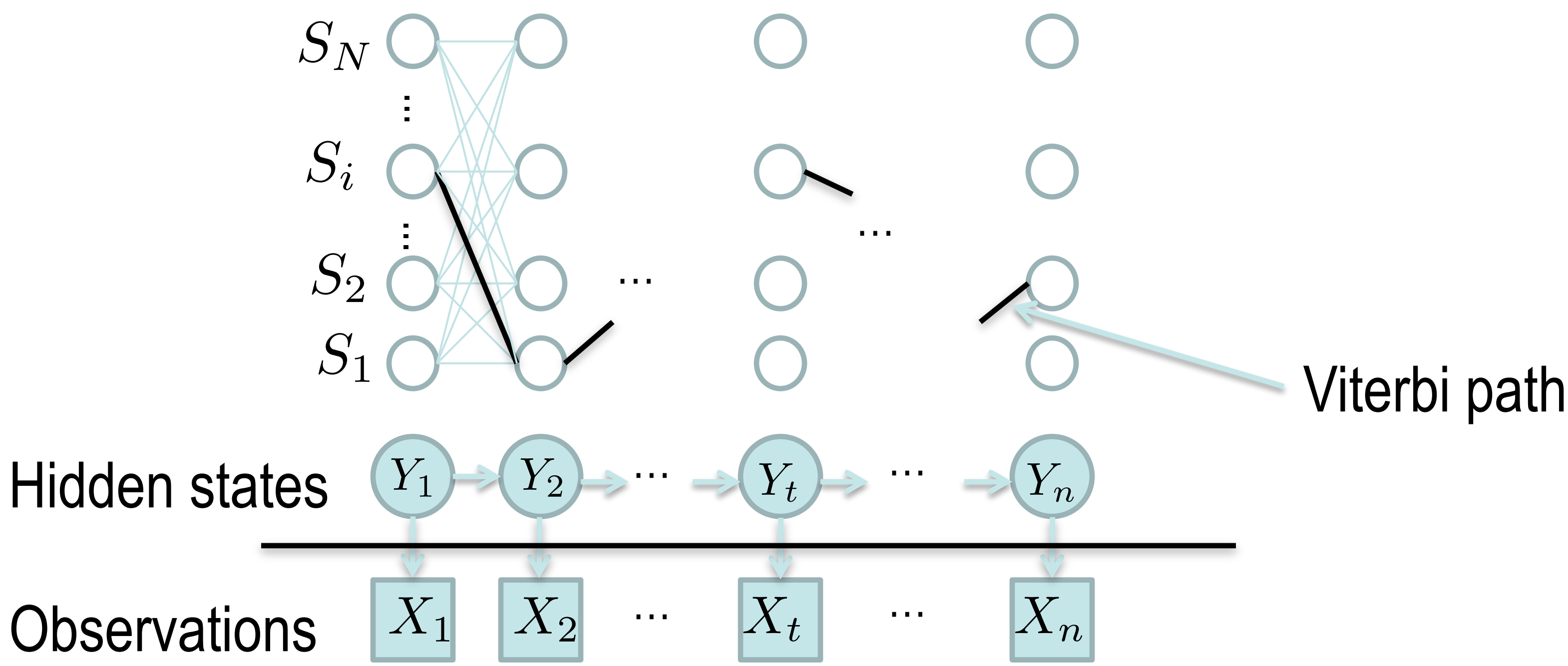
- Applications:
  - DP is a major paradigm in solving optimization problems
  - Viterbi algorithm (e.g., for hidden Markov models)
  - Cocke-Younger-Kasami (CYK) algorithm
  - Earley algorithm (a type of chart parser)
  - Value Iteration (e.g., for Markov decision process)
  - …

Four Steps in Developing a DP Algorithm

- 1. Characterize the structure of an optimal solution
- 2. Recursively define the value of an optimal solution
- 3. Compute this value in a bottom-up fashion
- 4. Find an optimal solution from computed information

DP for a Chain Model -- The Viterbi algorithm for HMM

- The goal: Find the most likely sequence of hidden states that produces the sequence of observed events



Hidden Markov Model (HMM)

- State space: $S = (S_1, \ldots, S_N)$, N states
- Distinct observation symbols: $V = (v_1, \ldots, v_M)$, M symbols
- Observation sequence: $\{X_1, X_2, \ldots, X_t, ...\}$, $X_t \in V$
- Hidden state sequence: $\{Y_1, Y_2, \ldots, Y_t, \ldots\}$, $Y_t \in S$
- State transition matrix: $A = (a_{ij})_{N\times N}$,

$$a_{ij} = p(Y_{t+1} = S_j | Y_t = S_i), \quad 1 \leq i, j \leq N$$

- Emission probability in state $S_j : B = (b_j(k))$,

$$b_j(k) = p(X_t = v_k | Y_t = S_j), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M$$

Hidden Markov Model (HMM)

- The prior initial state distribution: $\Pi_1 = (\pi_{11}, \pi_{12}, \ldots, \pi_{1N})$

$$\pi_{1j} = p(Y_1 = S_j), \quad 1 \leq j \leq N$$

- Joint probability:

$$\begin{aligned} p(X_1, \ldots, X_n, Y_1, \ldots, Y_n; A, B, \Pi_1) \\ = p(Y_1)p(X_1|Y_1) \prod_{t=2}^{n} p(Y_t|Y_{t-1})p(X_t|Y_t) \end{aligned}$$

Three Basic Problems in HMM

Hidden states $\mathbb{Y} = [Y_1, \ldots, Y_n]$

Observations $\mathbb{X} = [X_1, \ldots, X_n]$

Model parameters $\Theta = (A, B, \Pi_1)$

- Problem I: Given $\mathbb{X}$ and Θ , how to predict $\mathbb{Y}$? (i.e. Inference)

$$\mathbb{Y}^* = \arg\max_{\mathbb{Y}\in\Omega} p(\mathbb{Y}|\mathbb{X};\Theta) = \arg\max_{\mathbb{Y}\in\Omega} p(\mathbb{Y},\mathbb{X};\Theta)$$

where Ω is the solution space and $|\Omega| = N^n$

- Problem II: Given $\mathbb{X}$, how to compute the likelihood of model parameters,

$p(\mathbb{X};\Theta) =?$ (i.e. Membership)

- Problem III: How to estimate Θ based on $\mathbb{X}$? (i.e. Learning)

$$\widehat{\Theta}_{MLE} = \arg\max_{\Theta} p(\mathbb{X};\Theta)$$

The Viterbi Algorithm for Solving Problem I

- 1. Characterize the structure of an optimal solution
- 2. Recursively define the value of an optimal solution
- 3. Compute this value in a bottom-up fashion
- 4. Construct an optimal solution from computed information

The Viterbi Algorithm for Solving Problem I

- 1. Characterize the structure of an optimal solution

$$p^* = \max_{\mathbb{Y}} p(\mathbb{X}, \mathbb{Y}; \Theta) \quad \text{and} \quad \mathbb{Y}^* = \arg\max_{\mathbb{Y}} p(\mathbb{X}, \mathbb{Y}; \Theta)$$

- 2. Recursively define the value of an optimal solution

Denote $\mathbb{Y}_t = [Y_1, \ldots, Y_t]$ and $\mathbb{X}_t = [X_1, \ldots, X_t]$

Define $\delta_t(i) = \max_{\mathbb{Y}_{t-1}} p(\mathbb{Y}_{t-1}, Y_t = S_i, \mathbb{X}_t; \Theta)$

=> $\delta_{t+1}(j) = \max_{S_i} \; [\delta_t(i) \; a_{ij}] \; b_j(X_{t+1})$

=> $p^* = \max_{1 \le i \le N} \delta_n(i)$

The Viterbi Algorithm for Solving Problem I

- 3. Compute the value of an optimal solution in a bottom-up fashion

$$1 \leq i \leq N$$

$$\delta_1(i) \;=\; p(Y_1 = S_i, X_1; \Theta) \;=\; p(Y_1 = S_i)p(X_1 | Y_1 = S_i) \;=\; \pi_{1i} b_i(X_1)$$

- 4. Construct an optimal solution from computed information

The Viterbi Algorithm for Solving Problem I

- (1) Initialization:

$$\delta_1(i) = \pi_{1i} b_i(X_1),\ \ \gamma_1(i) = 0,\ \ 1 \le i \le N,$$

- (2) Forward maximization: for $t = 2, \ldots, n$,

$$\delta_t(i) = \max_{1 \le i \le N} \delta_{t-1}(i) a_{ij} b_j(X_t),\ \ 1 \le j \le N$$

$$\gamma_t(j) = \arg\max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}$$

- (3) Termination:

$$p^* = \max_{1 \le i \le N} \ \delta_n(i) \qquad\qquad Y_n^* = \arg\max_{1 \le i \le N} \ \delta_n(i)$$

- (4) Backward tracking: for $t = n-1, \ldots, 1$

$$Y_t^* = \gamma_{t+1}(Y_{t+1}^*)$$

The Viterbi Algorithm for Solving Problem I

- The time complexity of parsing the entire state sequence:

$$O(n \times N^2)$$

Forward-Backward Summation for Solving Problem II

- How to compute the likelihood $p(\mathbb{X};\Theta)$? (i.e. Membership)

$$p(\mathbb{X};\Theta) = \sum_{i=1}^{N} p(\mathbb{X}, Y_n = S_i; \Theta)$$

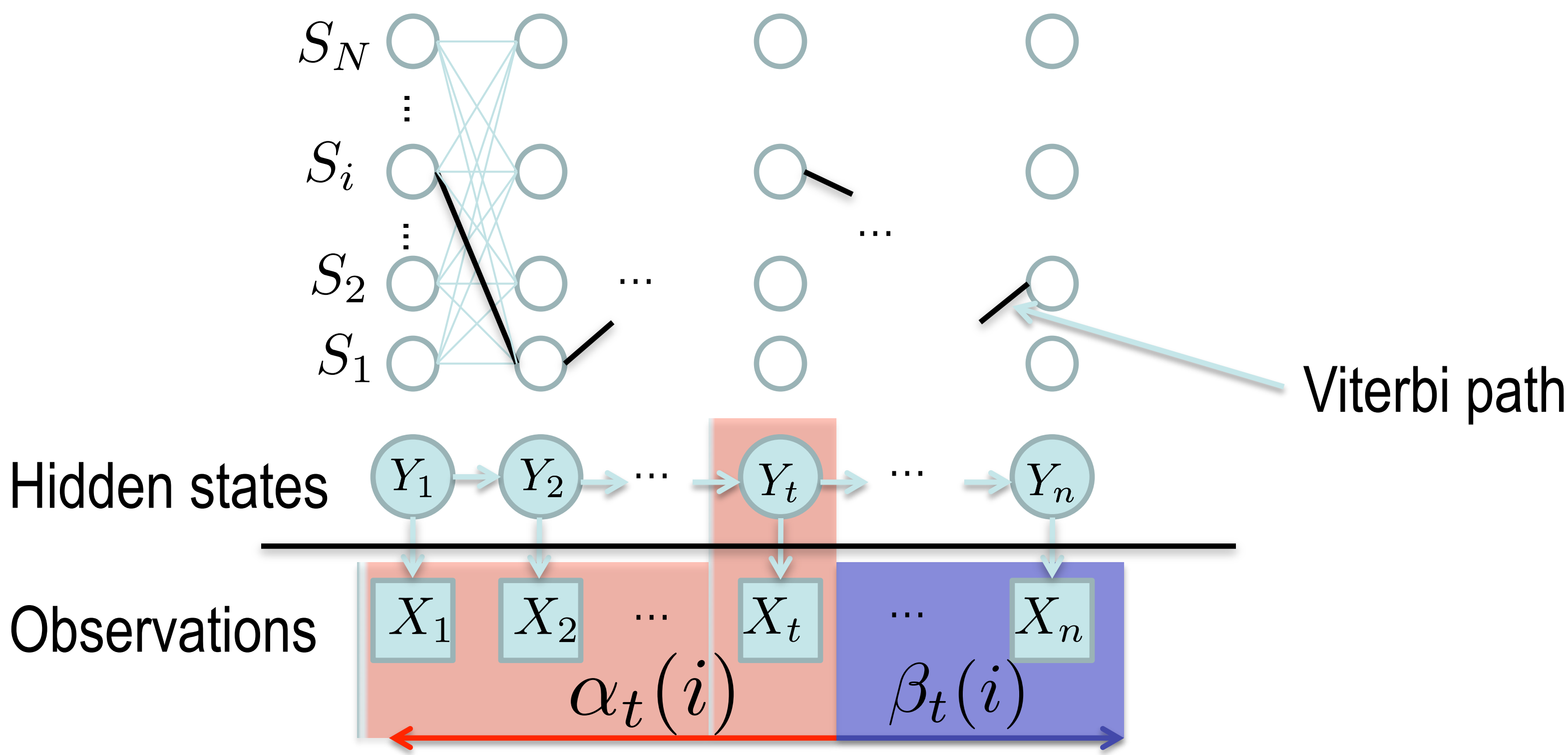
- How to compute the marginal belief at t $p(Y_t = S_i|\mathbb{X},\Theta)$?

$$p(Y_t = S_i|\mathbb{X};\Theta) = \frac{p(\mathbb{X}, Y_t = S_i; \Theta)}{p(\mathbb{X};\Theta)}$$

Forward-Backward Summation for Solving Problem II

- Define $\alpha_t(i) = p(\mathbb{X}_t, Y_t = S_i; \Theta)$, where $\mathbb{X}_t = [X_1, \ldots, X_t]$

  $\beta_t(i) = p(\mathbb{X}_{-t} | Y_t = S_i; \Theta)$, where $\mathbb{X}_{-t} = [X_{t+1}, \ldots, X_n]$



Forward-Backward Summation for Solving Problem II

- Define $\alpha_t(i) = p(\mathbb{X}_t, Y_t = S_i; \Theta)$, where $\mathbb{X}_t = [X_1, \ldots, X_t]$

  $\beta_t(i) = p(\mathbb{X}_{-t} | Y_t = S_i; \Theta)$, where $\mathbb{X}_{-t} = [X_{t+1}, \ldots, X_n]$

- Then, $\alpha_1(i) = \pi_{1i} b_i(X_1)$

  $\beta_n(i) = 1$ // empty string, so probability = 1

=> $$p(\mathbb{X}; \Theta) = \sum_{i=1}^{N} p(\mathbb{X}, Y_n = S_i; \Theta) = \sum_{i=1}^{N} \alpha_n(i)$$

$$p(Y_t = S_i | \mathbb{X}; \Theta) = \frac{p(\mathbb{X}, Y_t = S_i; \Theta)}{\sum_{j=1}^{N} p(\mathbb{X}, Y_t = S_j; \Theta)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

(used later in Inside/Outside)

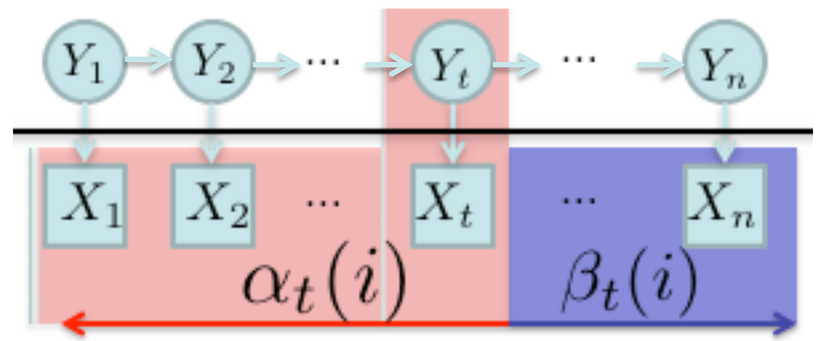
Forward and Backward Recursions

- Forward recursion:

$$\alpha_{t+1}(j) = p(\mathbb{X}_{t+1}, Y_{t+1} = S_j; \Theta)$$

$$= \sum_{i=1}^{N} p(\mathbb{X}_t, X_{t+1}, Y_t = S_i, Y_{t+1} = S_j; \Theta)$$

$$= \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(X_{t+1})$$



- Backward recursion:

$$\beta_t(i) = p(\mathbb{X}_{-t} | Y_t = S_i; \Theta)$$

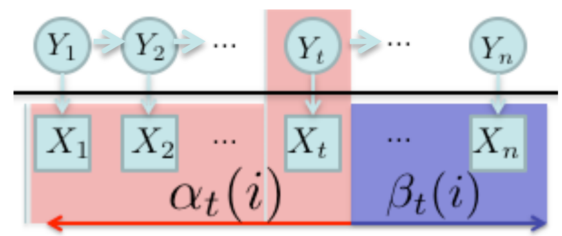
$$= \sum_{j=1}^{N} p(\mathbb{X}_{-t}, Y_{t+1} = S_j | Y_t = S_i; \Theta)$$

$$= \sum_{j=1}^{N} \alpha_{ij} b_j(X_{t+1}) \beta_{t+1}(j)$$

The Forward-Backward Summation Algorithm

- The forward summation
  - (1) Initialization $\alpha_1(i) = \pi_{1i} b_i(X_1) \quad 1 \leq i \leq N$
  - (2) Recursion: for $t = 1, 2, \ldots, n-1$

    $1 \leq j \leq N,\ \alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(X_{t+1})$
  - (3) Termination $p(\mathbb{X}; \Theta) = \sum_{i=1}^{N} \alpha_n(i)$
- The backward summation



  - (1) Initialization $\beta_n(i) = 1,\ 1 \leq i \leq N$
  - (2) Recursion: for $t = n-1, n-2, \ldots, 1$

    $1 \leq i \leq N,\ \beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)$
  - (3) $1 \leq t \leq n,\ 1 \leq i \leq N \quad p(Y_t = S_i | \mathbb{X}; \Theta) = \dfrac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$

Chart Parsing

- Motivation: General search not suitable

Local ambiguities of grammar => The same syntactic constituent may be rederived as a part of larger constituents

- Basic idea: Do not throw away any information. Keep a record --- a chart --- of all the structures found

Two Types of Chart Parsing

- Passive = Bottom-up parsing
- Active = Agenda-driven chart parsing
  - Bottom-up active chart parsing
  - Top-down active chart parsing
  - Agenda is used to prioritize constituents to be processed as
    - Stack to simulate depth-first search (DFS)
    - Queue to simulate breadth-first search (BFS)
    - Priority queue to simulate best-first search

What is a chart?

Chart = Well-formed substring table (WFST)

- Plays a role of the memo-table as in DP
- Keeps a track of partial derivations

What is a chart?

Charts are represented by directed graphs $G = (V, E)$

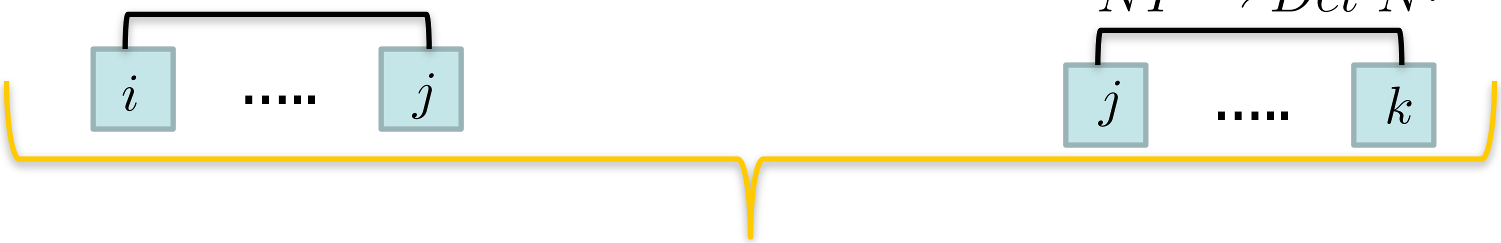
- $V = \{1, 2, \ldots, n\}$ represents the input sentence, where i - th node corresponds to i -th word,
- Each edge $e \in E$ represents a completed, or partial constituent which spans a group of words, e.g., $e = (start, end, label, found, tofind) \in E$
  - $label$ = Nonterminal node, e.g., LHS of a certain rule in grammar
  - $found$ = Part of RHS of $label$ which explains words from $start$ to end
  - $tofind$ = Remainder of the sentence beside the $found$ part
  - Active edge: $tofind$ is not empty
  - Inactive edge (passive edge): $tofind$ is empty

The Fundamental Rule: Combines active and passive edges

$$e = (start, end, label, found, tofind) \in E$$

An active edge: $e_1 = (i, j, VP, DV, NP\ PP)$ A passive edge: $e_2 = (j, k, NP, Det\ N, \emptyset)$

$VP \rightarrow DV \cdot NP\ PP$ $NP \rightarrow Det\ N \cdot$



$$VP \rightarrow DV\ Det\ N \cdot PP$$

The fundamental rule

The Fundamental Rule: Combines active and passive edges

$$e = (start, end, label, found, tofind) \in E$$

An active edge: $e_1 = (i, j, VP, DV, NP\ PP)$ A passive edge: $e_2 = (j, k, NP, Det\ N, \emptyset)$

$$VP \rightarrow DV\ Det\ N \cdot PP$$

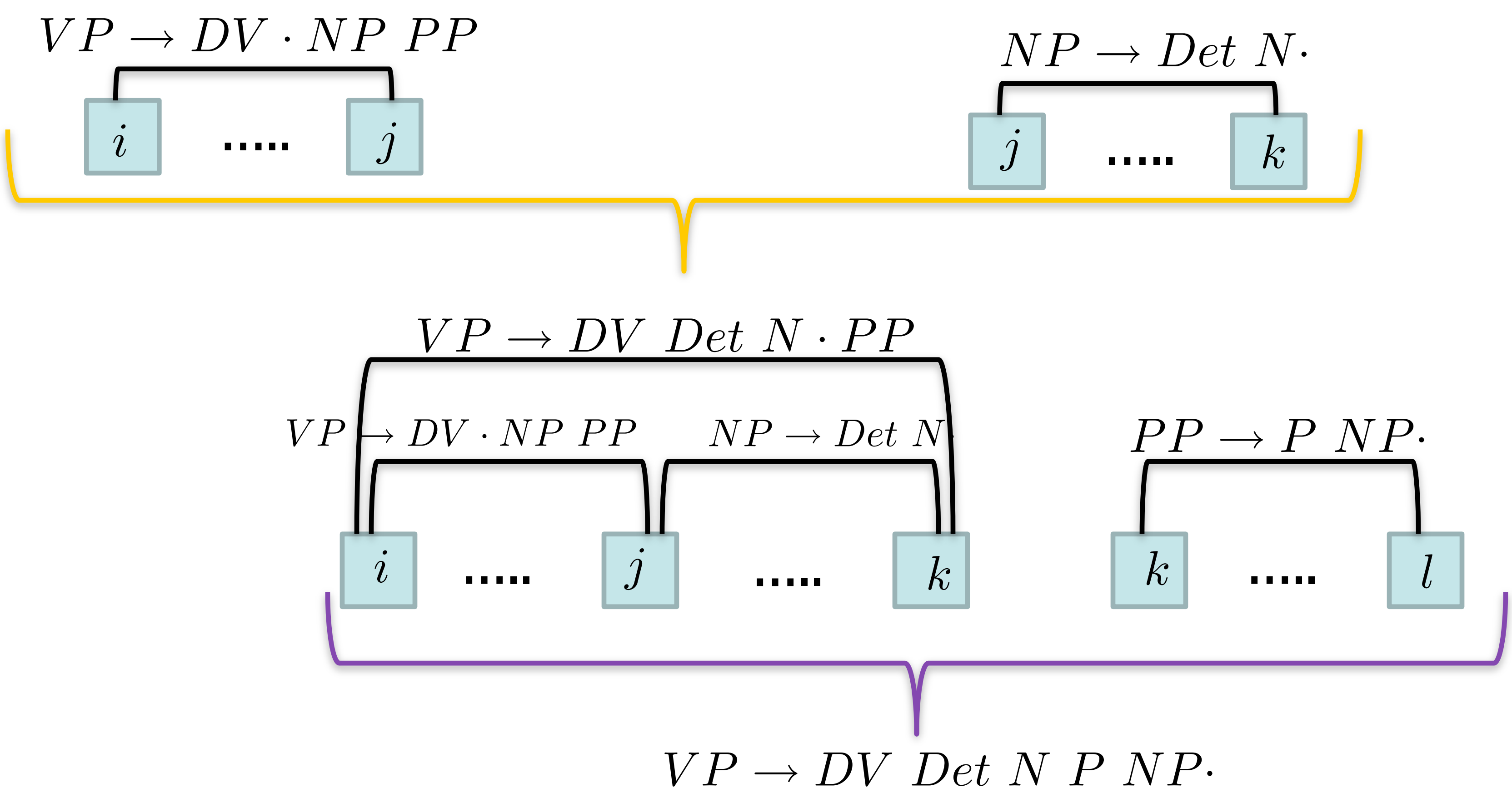
We get a new active edge

$$e_3 = (i, k, VP, DV\ Det\ N, PP)$$

The Fundamental Rule: Combines Active and Passive Edges

$$e = (start, end, label, found, tofind) \in E$$

An active edge: $e_1 = (i, j, VP, DV, NP\ PP)$ A passive edge: $e_2 = (j, k, NP, Det\ N, \emptyset)$

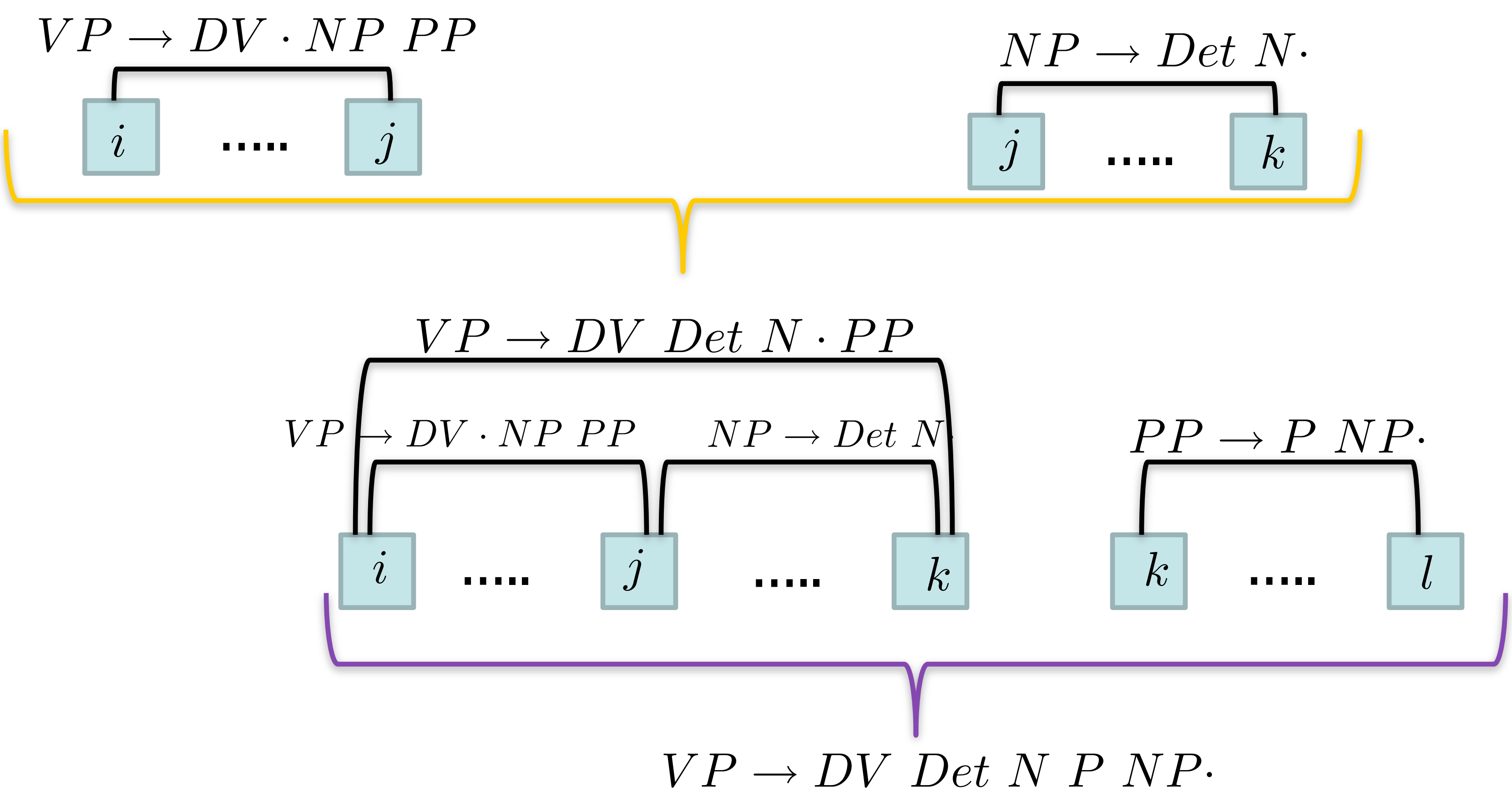


The fundamental rule

The Fundamental Rule: Combines Active and Passive Edges

$$e = (start, end, label, found, tofind) \in E$$

An active edge: $e_1 = (i, j, VP, DV, NP\ PP)$ A passive edge: $e_2 = (j, k, NP, Det\ N, \emptyset)$

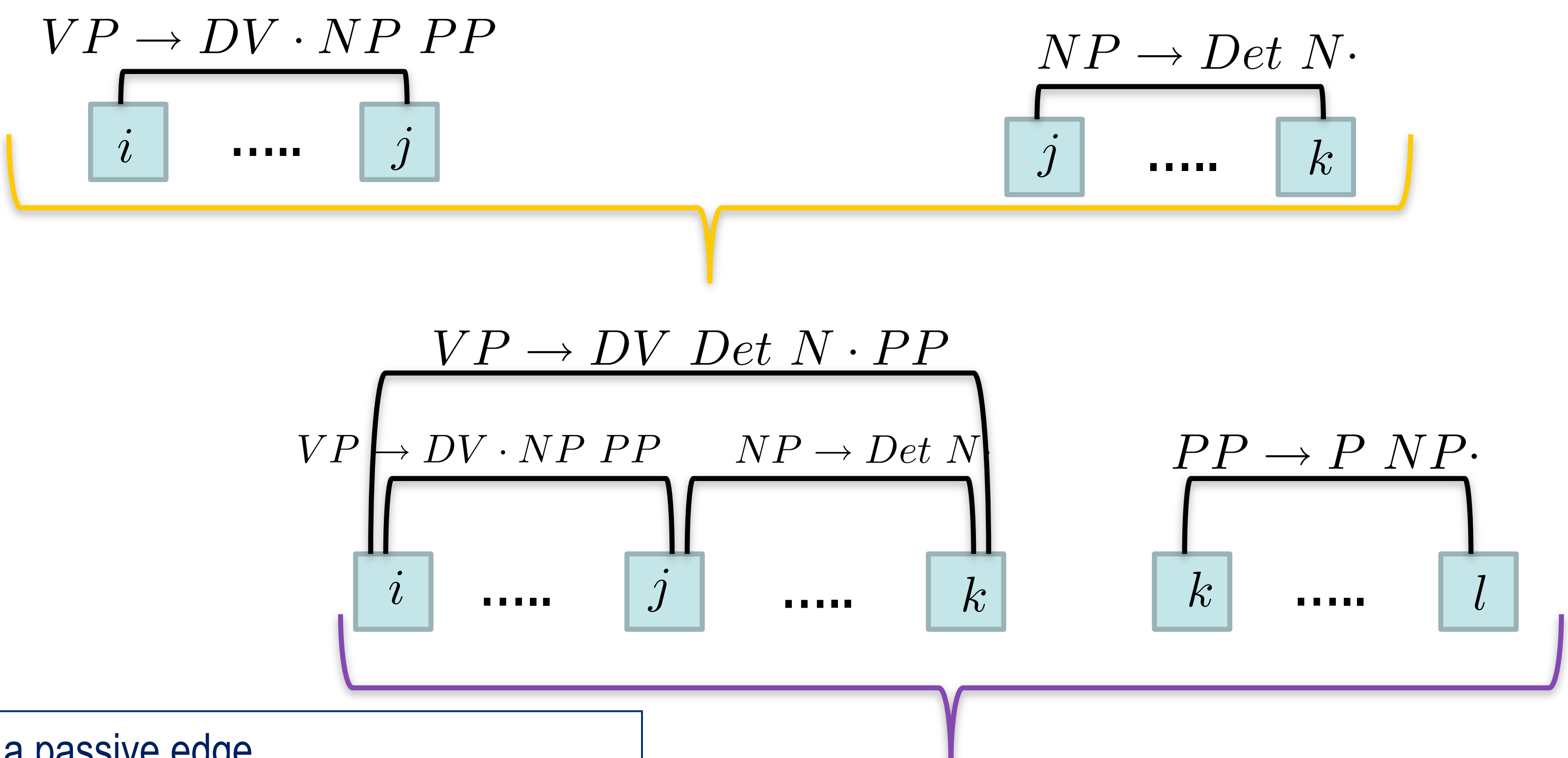


The fundamental rule

The Fundamental Rule: Combines Active and Passive Edges

$$e = (start, end, label, found, tofind) \in E$$

An active edge: $e_1 = (i, j, VP, DV, NP\ PP)$ A passive edge: $e_2 = (j, k, NP, Det\ N, \emptyset)$



$$VP \rightarrow DV\ \ Det\ N\ P\ NP \cdot$$

The fundamental rule

We get a passive edge

$$e_4 = (i, l, VP, DV\ \ Det\ N\ \ PP, \emptyset)$$

What is a agenda?

- Agenda = Set of edges waiting to be added to the chart

- Determines the order in which 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

- Ordering is decided by Figures of Merit (FOM) of elements

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:

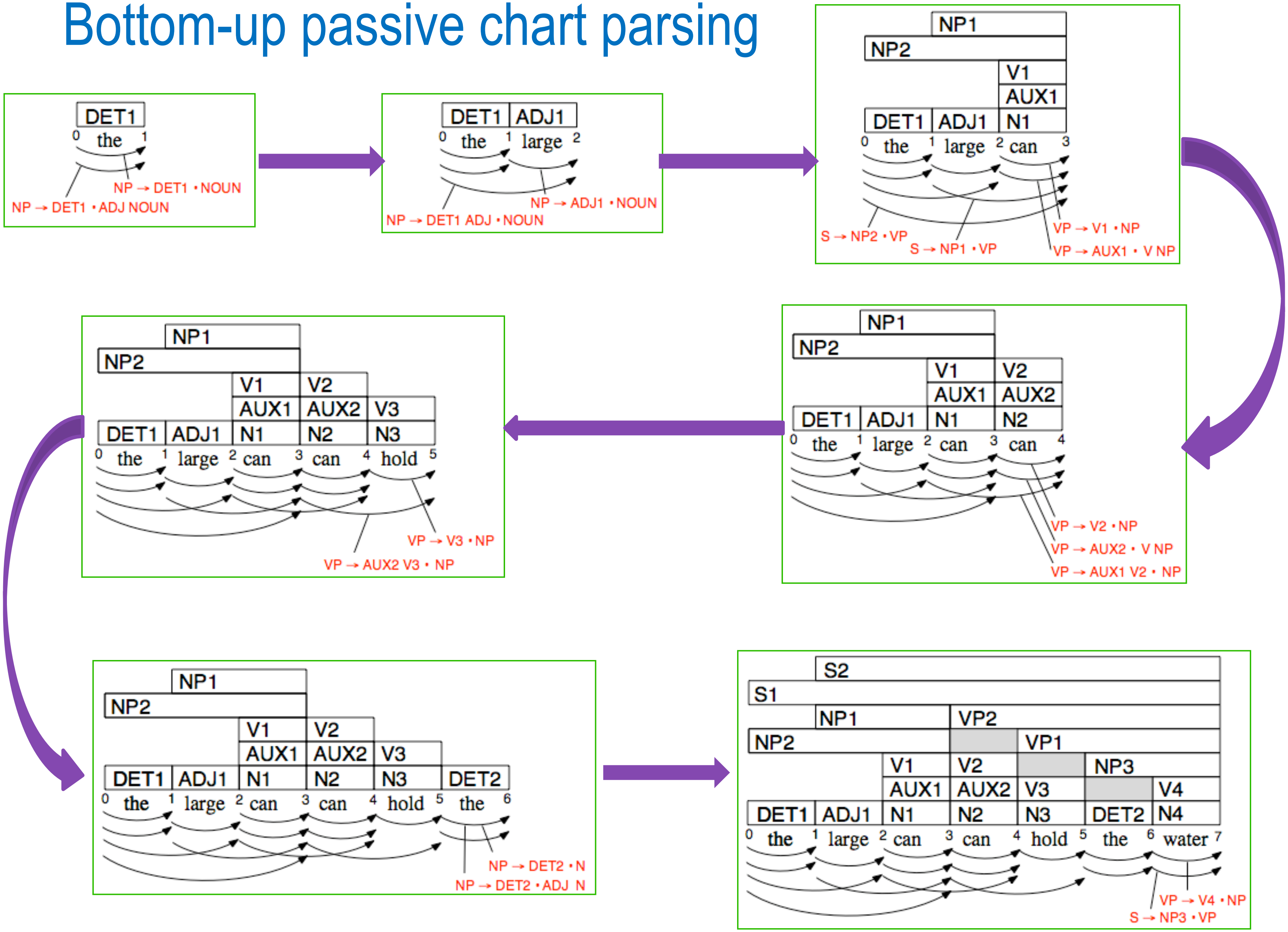
the... DET
large... ADJ
can... AUX, N, V
hold... N, V
water... N, V

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

Cocke–Younger–Kasami (CYK) algorithm

- CYK algorithm = Bottom-up passive chart parsing algorithm
- The context-free grammar (CFG) must be in Chomsky normal form (CNF)
- The goal:
  - Determine if the sentence can be generated by a given CFG
  - If so, how it can be generated (e.g., parse tree construction)

Bottom-up passive chart parsing



Cocke–Younger–Kasami (CYK) algorithm

- The worst case running time of CYK is $O(n^3|G|)$

  n = Length of the input sentence and

  $|G|$ = Size of grammar

- Drawback of all known transformations into CNF:

  May lead to a blow-up in grammar size

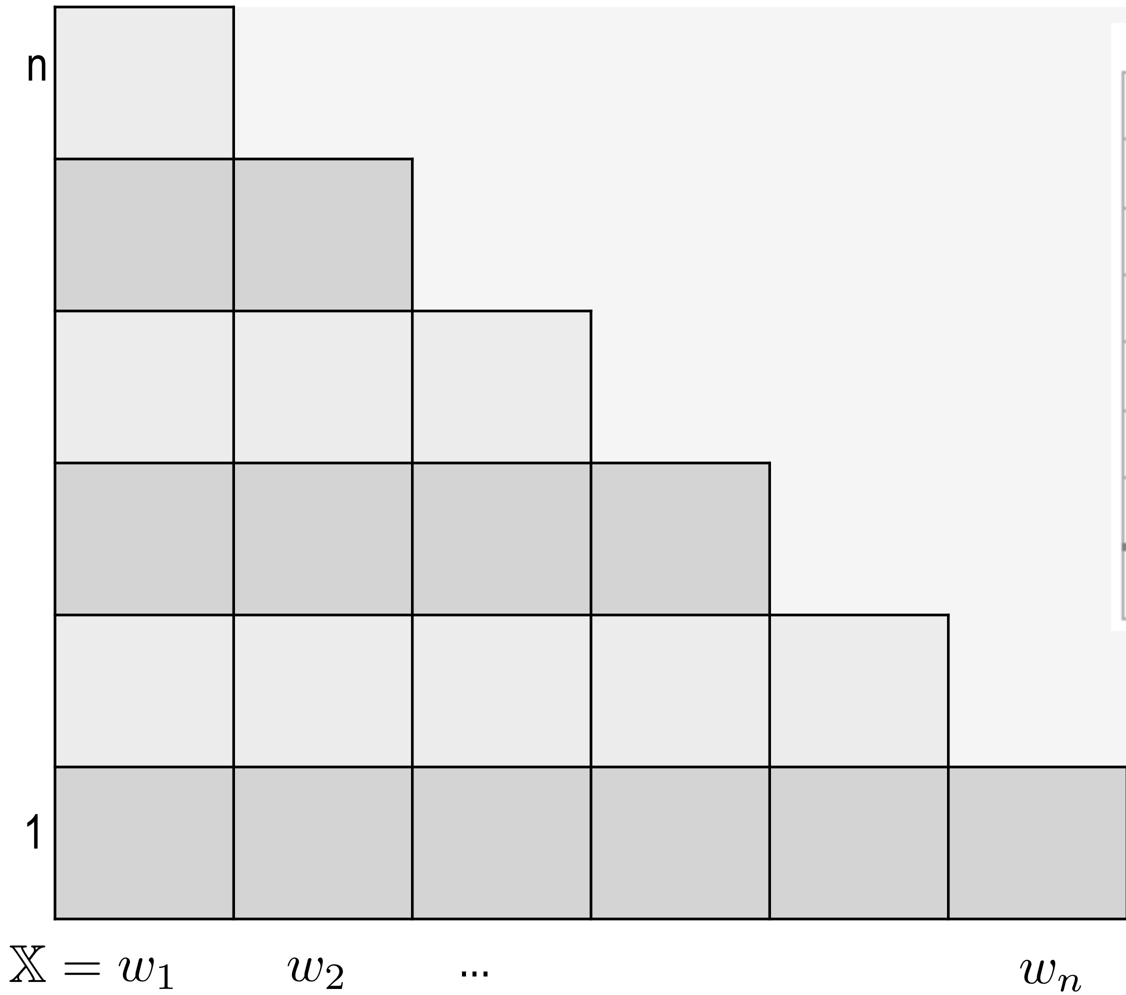
- Let g be the size of grammar

  => blow-up may range from g^2 to 2^{2g}

CYK algorithm

- Input: a sentence $\mathbb{X} = w_1 \ldots w_n$ and the grammar G with S being the root.
  - Let $w_{ij} = w_i w_{i+1} \ldots w_{i+j-1}$ be the substring of $\mathbb{X}$ of length j starting with w_i . Then, we have $\mathbb{X} = w_{1n}$.
- Output: verify whether $S \Rightarrow \mathbb{X}$. If yes, construct all possible parse trees.
- The algorithm: for every w_{ij} and every rule $R \in G$, determine if $R \Rightarrow w_{ij}$ and the probability if necessary.
  - Define an auxiliary 4-tuple variable for each rule $R_k \in G$:
    $v_k = (k,\ probability,\ pointerLeft,\ pointerRight)$
  - CYK table with the entries $V_{ij},\ 1 \leq i \leq n,\ 1 \leq j \leq n - i + 1$ storing the auxiliary variables of the rules which can explain substring .
  - Start with substrings of length 1: $w_{i1} = w_i,\ 1 \leq n$, set
    $$V_{i1} = \{v_k = (k, Pr(R_k|w_{i1}), \emptyset, \emptyset) : R_k \Rightarrow w_{i1},\ R_k \in G\}$$
  - Continue with substrings of length $j = 2, 3, \ldots, n - i + 1$
    - For w_{ij}, consider all two-part partitions $w_{ij} = w_{im} w_{i+m\ j-m},\ 1 \leq m \leq j$
      $$V_{ij} = \{v_k = (k, Pr(R_k|w_{ij}), v_{k_l}, v_{k_r}) : R_k \Rightarrow R_{k_l} R_{k_r},\ R_{k_l} \Rightarrow w_{im},\ R_{k_r} \Rightarrow w_{i+m\ j-m},\ R_k, R_{k_l}, R_{k_r} \in G\}$$

CYK algorithm -- 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

Bottom-up Active Chart Parsing – Algorithm Flow

1. Initialize chart and agenda

   Chart = empty, Agenda = {passive edges for all rules for all words}

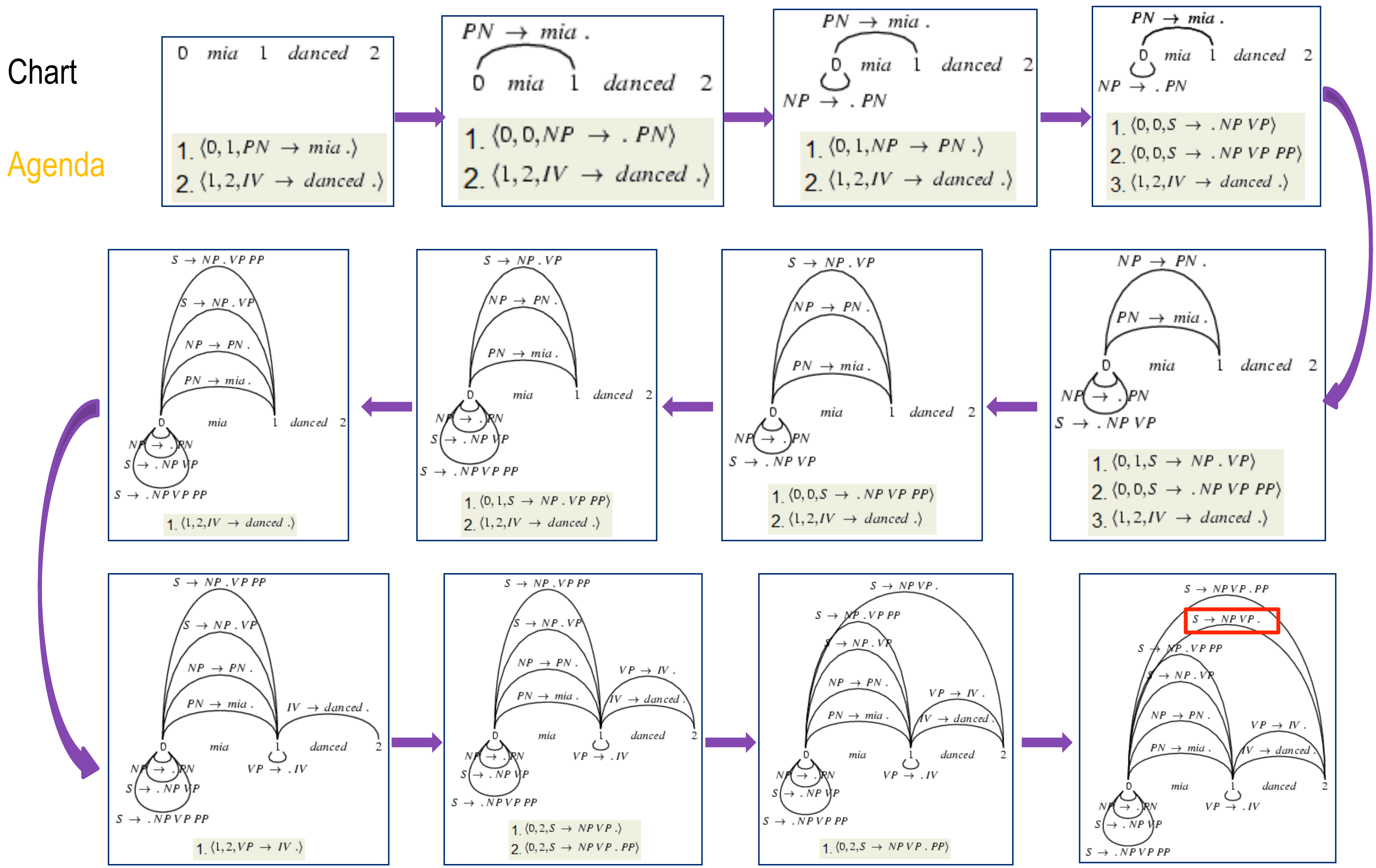
2. Repeat until agenda is empty
   - Select an edge from Agenda (e.g., DFS, BFS)
     $e = (start, end, label, found, tofind) \in E$
   - Add e to the chart at position $(start,\ end)$ if it is not in the chart
   - Use the fundamental rule to combine e with other edges from the chart
   - If e is PASSIVE, look for grammar rules r which have $found$ as the first symbol on the RHS
   - For each r , build active edge e' and add it to Agenda
     $e' = (start, start, r, \emptyset, found\ V_{remaining})$

3. Succeed if there is a passive edge $e = (0, n, S, found, 0)$, where S is the root node in grammar

An example of bottom-up 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



Top-down vs. Bottom-up Active Chart Parsing

- Bottom-up active chart parsing
  - Checks the input sentence, and builds each constituent exactly once. No duplication of effort.
  - May build constituents that cannot be later used legally
  - Reads the rules right-to-left, and starts with the information in passive edges
- Top-down chart parsing
  - Highly predictive. Only grammar rules that can be legally applied will be put to the chart
  - Reads the rules left-to-right, and starts with the information in active edges

Top-down Active Chart Parsing – Earley Parser

- Initialize chart and agenda

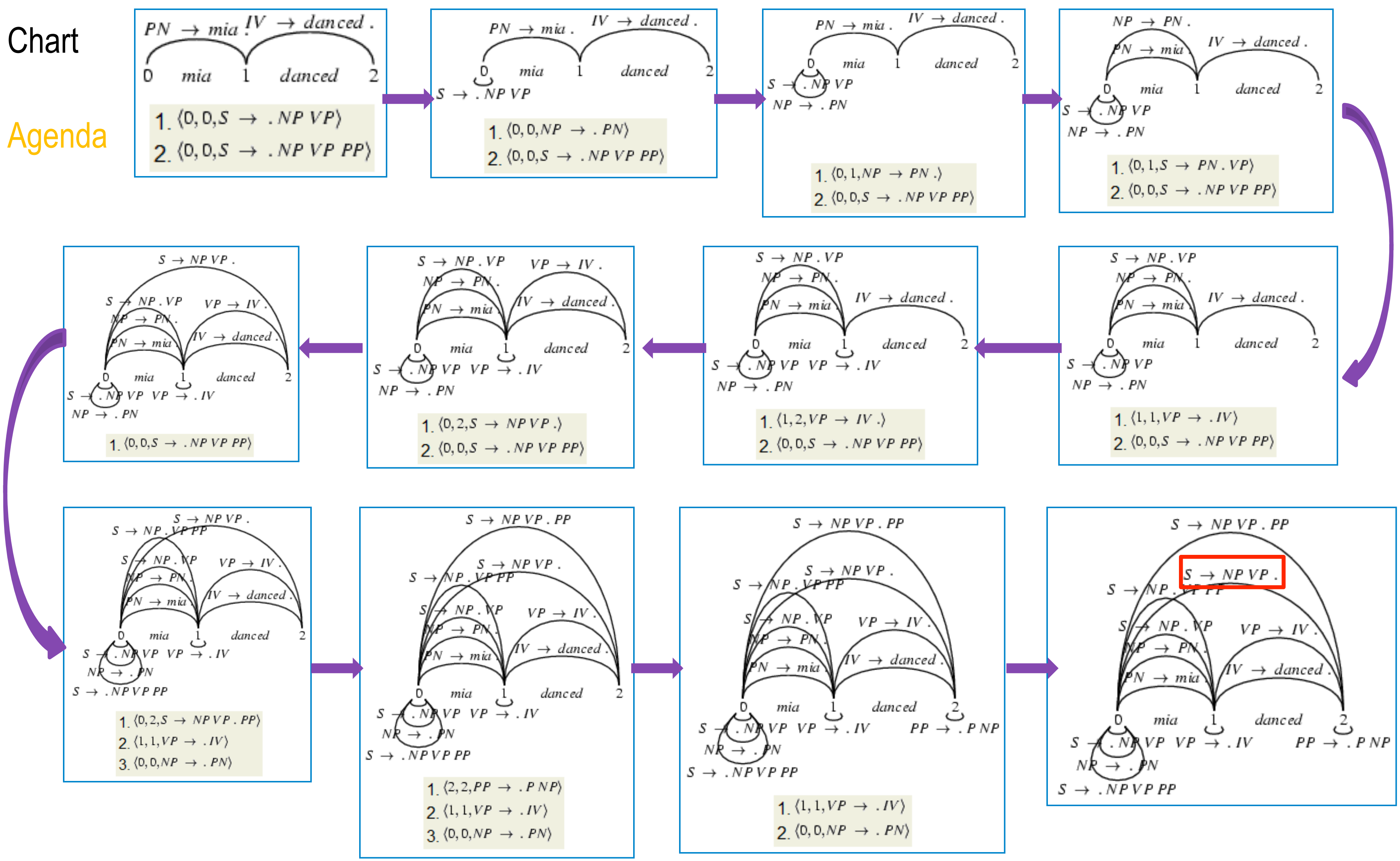
  Chart = {passive edges for all rules for all words}, Agenda = {root rules}
- Repeat until agenda is empty
  - Select an edge from Agenda (e.g., DFS, BFS)
    $e = (start, end, label, found, tofind) \in E$
  - Add e to the chart at position $(start,\ end)$ if it is not in the chart
  - Use the fundamental rule to combine e with other edges from the chart
  - If e is ACTIVE, then look for grammar rules r which have the form
    $r =\ tofind \rightarrow V_1 \ldots V_m$
  - For each r, build active edge e' and add it to Agenda

    $$e' = (end, end, r, \emptyset, V_1 \ldots V_m)$$
- Succeed if there is a passive edge $e = (0, n, S, found, \emptyset)$, where S is the root node in grammar

An example of top-down active chart parsing

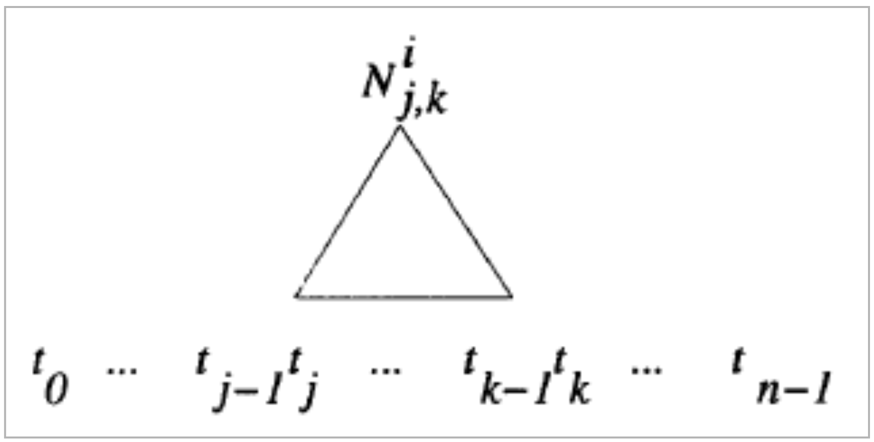
$S \rightarrow NP\ VP$
$S \rightarrow NP\ VP\ PP$
$NP \rightarrow PN$
$VP \rightarrow IV$
$PP \rightarrow P\ NP$
$PN \rightarrow mia$
$IV \rightarrow danced$

Mia danced



Best-first Chart Parsing

- Agenda uses a priority queue to keep track of partial derivations
- Ordering is calculated using figures of merit of constituents (FOM)
- FOM ≈ Likelihood that the constituents will appear in a correct parse



Constituent $N^i_{j,k}$ in the sentence

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

Key problem: How to estimate

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

Ideal Figures of Merit

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

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

Ideal Figures of Merit

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

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

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

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

Ideal Figures of Merit

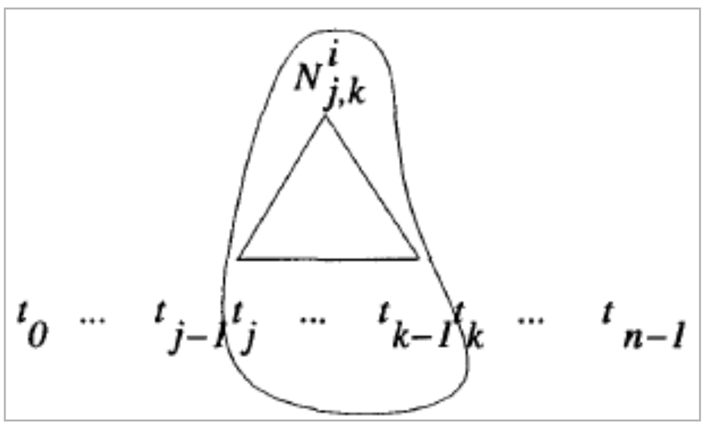
$$p(N^i_{j,k}|t_{0,n}) = \frac{p^{out}(N^i_{j,k})p^{in}(N^i_{j,k})}{p(t_{0,n})}$$

outside probability

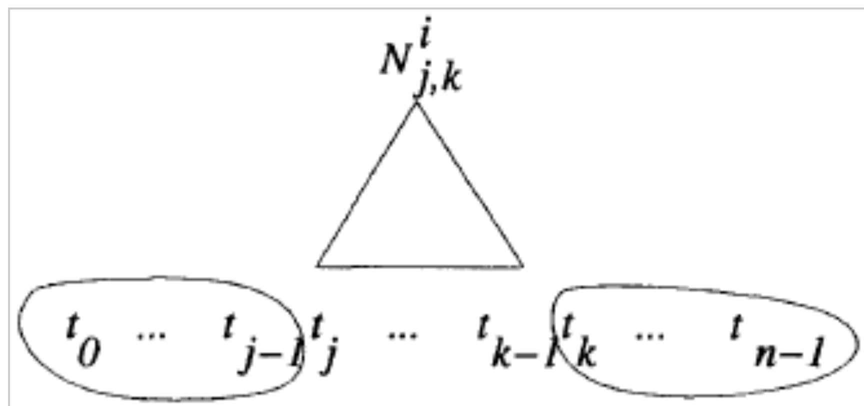
$$p^{out}(N^i_{j,k}) = p(N^i_{j,k}, t_{0,j}, t_{k,n})$$

inside probability

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



Inside probability includes only words within the constituent



Outside probability includes the entire context of the constituent

Simple Figures of Merit – Trigram Estimate

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

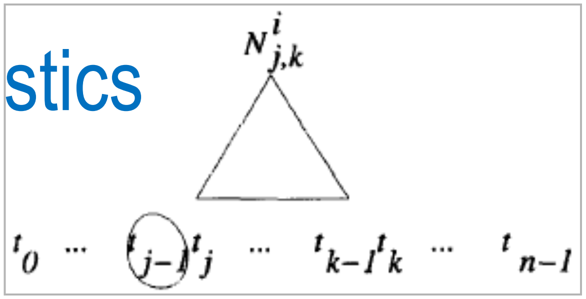
Assume

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

$$p(t_{j,k}|t_{0,j}, t_{k,n}) \approx p(t_{j,k}|t_{j-2}, t_{j-1}) = \prod_{a=j}^{k-1} p(t_a|t_{a-2}, t_{a-1})$$

=> $$p(N^i_{j,k}|t_{0,n}) \approx \frac{p(N^i)p^{in}(N^i_{j,k})}{\prod_{a=j}^{k-1} p(t_a|t_{a-2}, t_{a-1})}$$

Figures of Merit Using Boundary Statistics



Left boundary trigram estimate

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

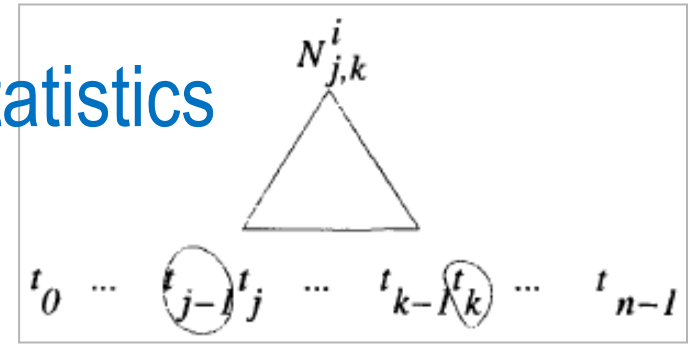
$$\approx \frac{p(N^i_{j,k}|t_{0,j}, t_{k,n})p^{in}(N^i_{j,k})}{p(t_{j,k}|t_{0,j}, t_{k,n})}$$

Assume: $p(t_{j,k}|t_{0,j}, t_{k,n}) \approx p(t_{j,k}|t_{j-2}, t_{j-1})$

$p(N^i_{j,k}|t_{0,j}, t_{k,n}) \approx p(N^i_{j,k}|t_{j-1})$ left boundary model

=> $$p(N^i_{j,k}|t_{0,n}) \approx \frac{p(N^i_{j,k}|t_{j-1})p^{in}(N^i_{j,k})}{p(t_{j,k}|t_{j-2}, t_{j-1})}$$

Figures of Merit Using Boundary Statistics



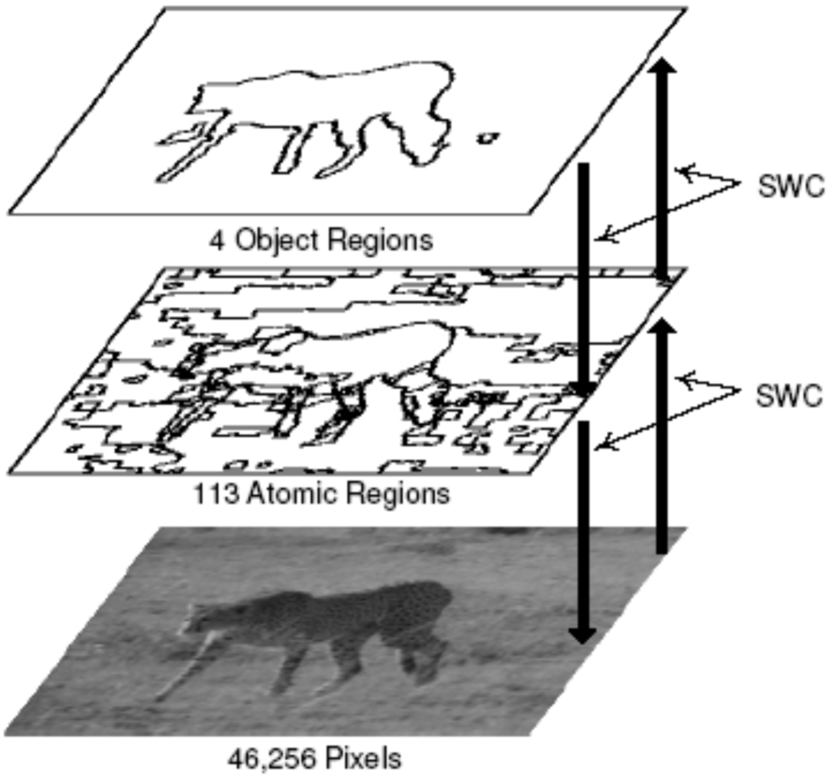
$$p(N^i_{j,k}|t_{0,n}) = \frac{p(N^i_{j,k}|t_{0,j})p(t_{j,k}|N^i_{j,k},t_{0,j})p(t_k|N^i_{j,k},t_{0,k})p(t_{k+1,n}|t_{0,k+1},N^i_{j,k})}{p(t_{0,j})p(t_{j,k}|t_{0,j})p(t_k|t_{0,k})p(t_{k+1,n}|t_{0,k+1})}$$

Assume $t_{k+1,n}$ depends only on the previous tags

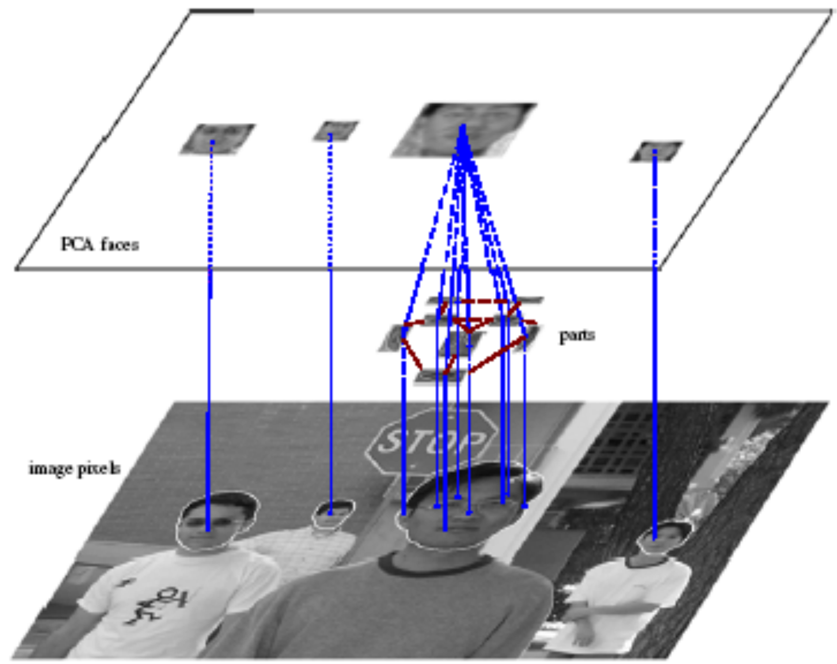
$$\Rightarrow \quad p(N^i_{j,k}|t_{0,n}) \approx \frac{p(N^i_{j,k}|t_{0,j})p^{in}(N_{j,k})p(t_k|N^i_{j,k},t_{0,k})}{p(t_{j,k+1}|t_{0,j})}$$

$$\approx \frac{p(N^i_{j,k}|t_{0,j})p^{in}(N_{j,k})p(t_k|N^i_{j,k})}{p(t_{j,k+1}|t_{j-2},t_{j-1})}$$

Open Problem: Bottom-up or Top-down Inference



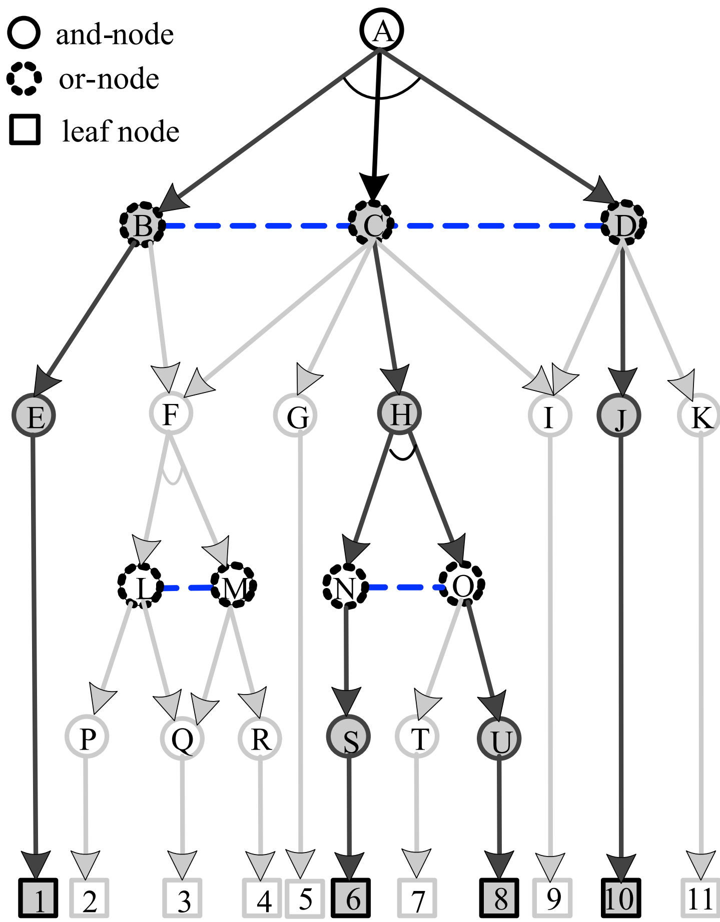
(a) bottom-up graph construction



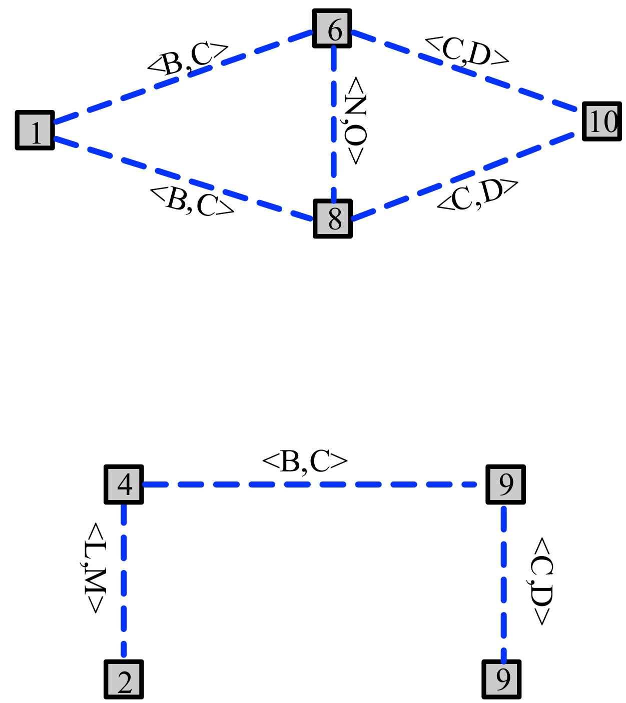
(b) Top-down graph construction

Which inference is more suitable?
This is object-dependent

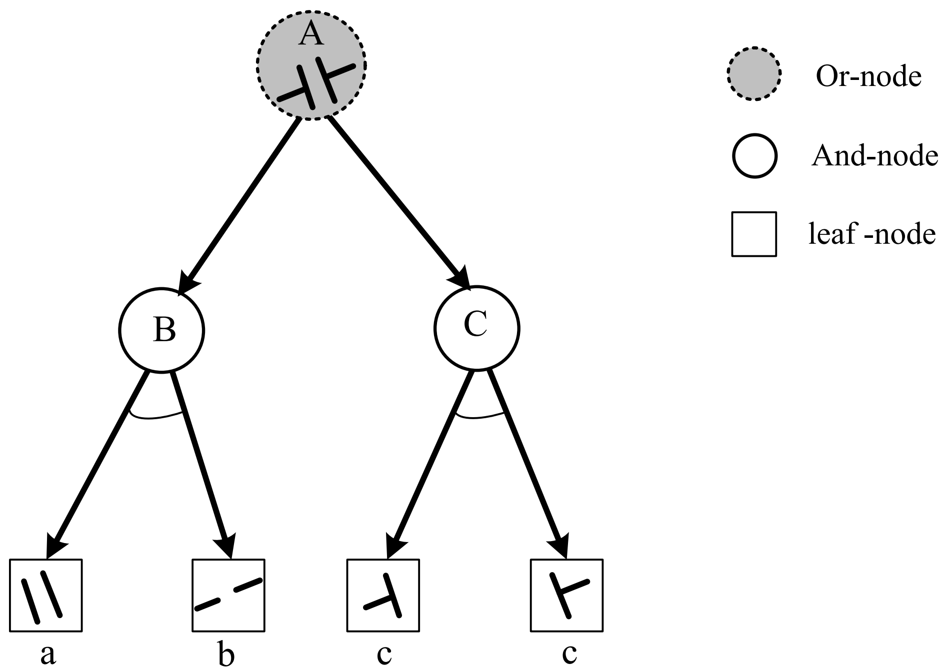
Embedding the integrated models into an And-Or graph



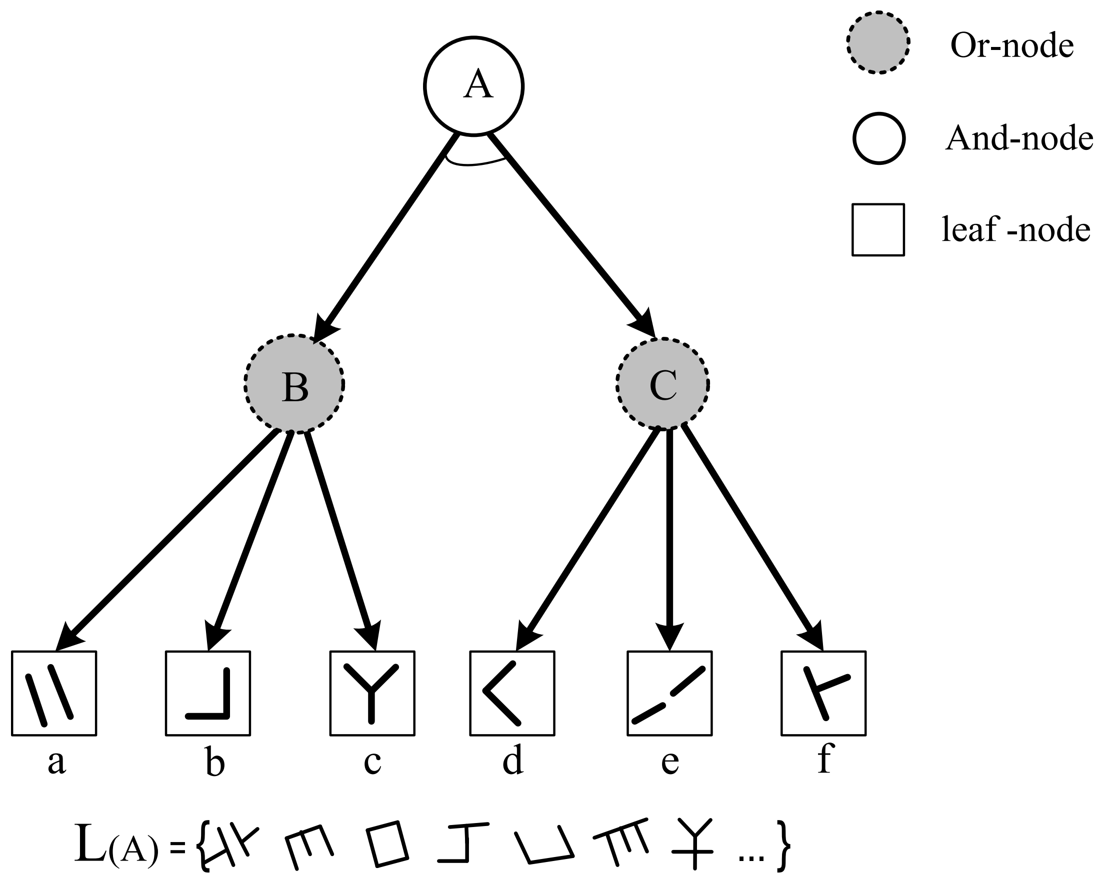
some graph configurations generated by the And-Or graph



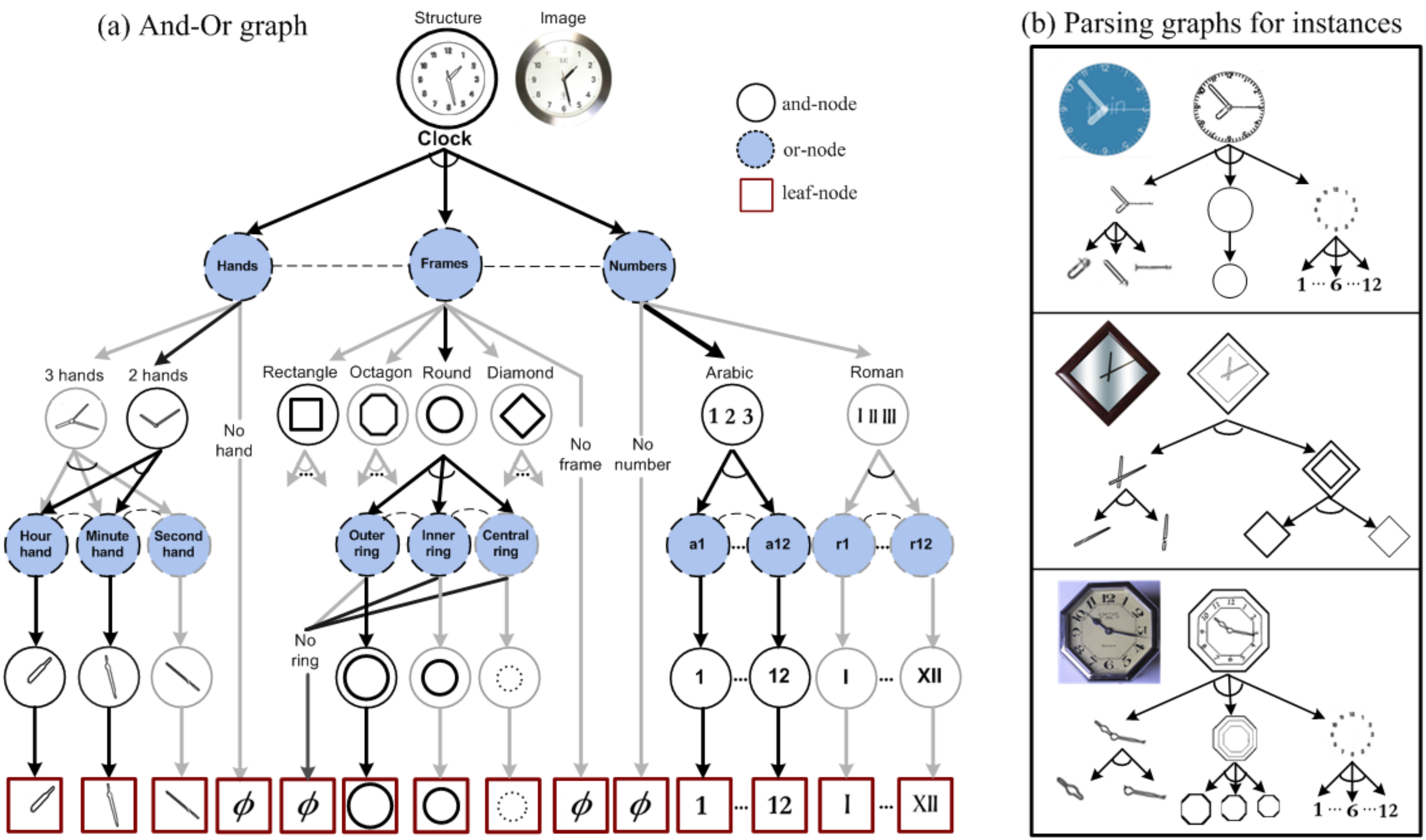
Representing a grammar by and-or graph



Representing a grammar by and-or graph



An example: the clock category



Top-down / Bottom-up Inference at all levels

Objective: Constructing parse graphs on-line !

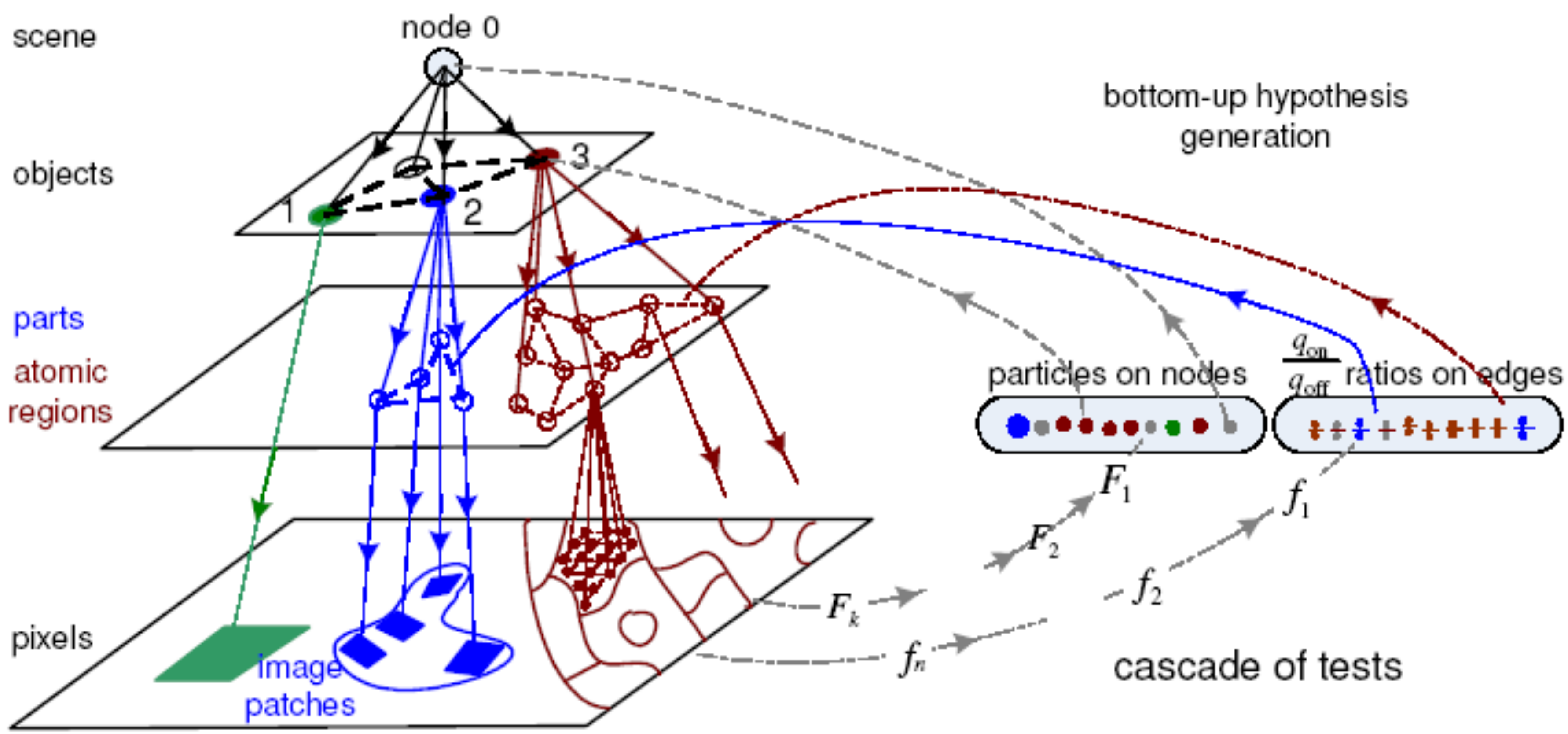
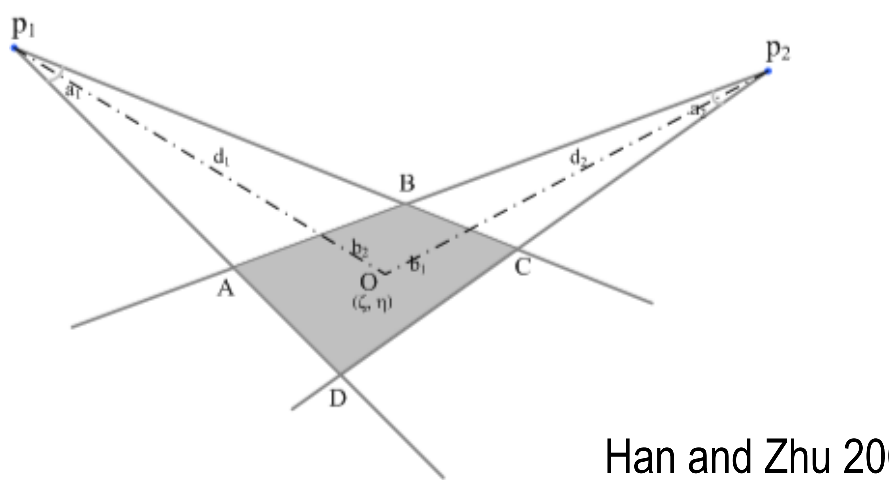


Image parsing by DDMCMC, Tu et al, 2002-05

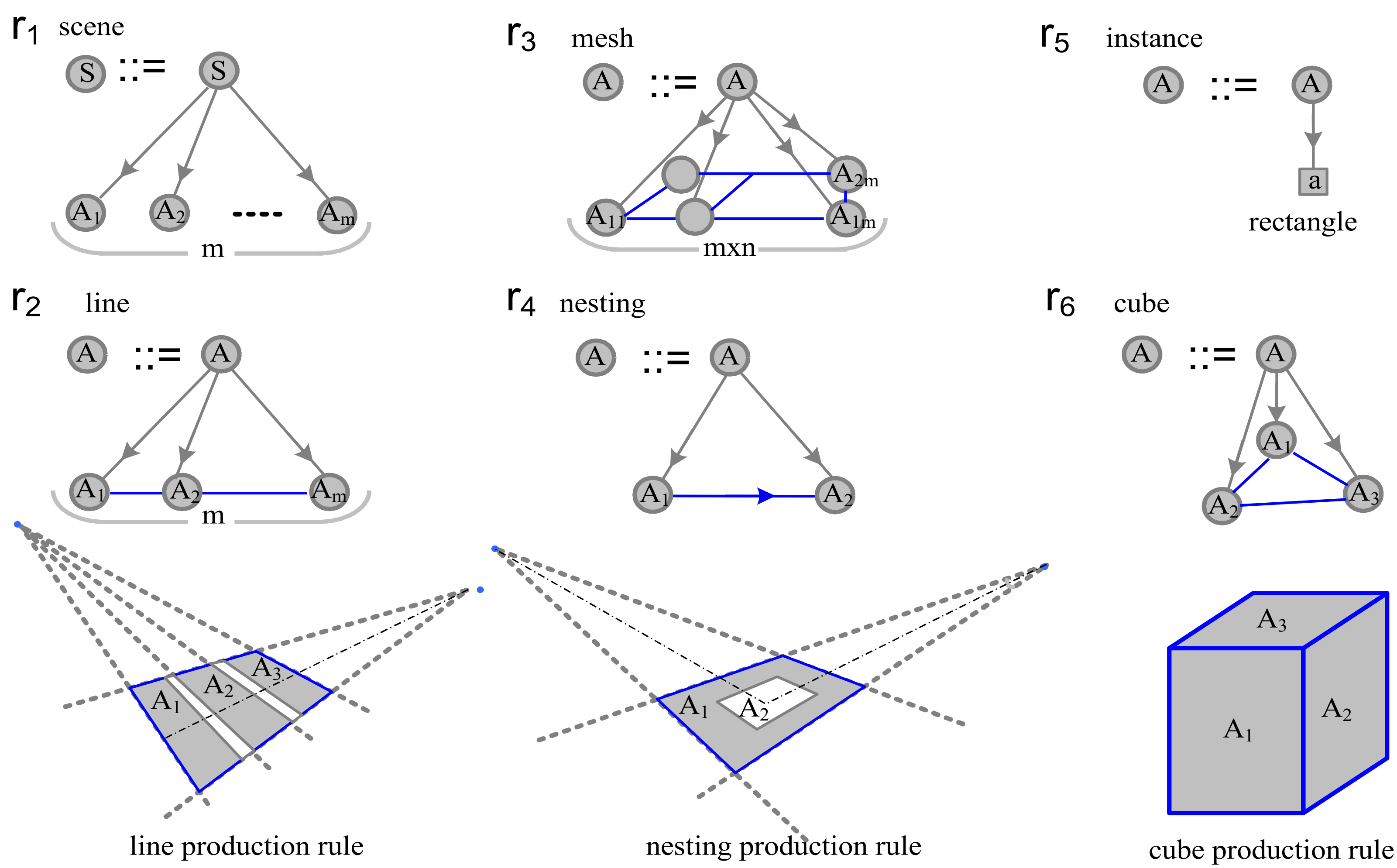
A simpler and more flexible graph grammar

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

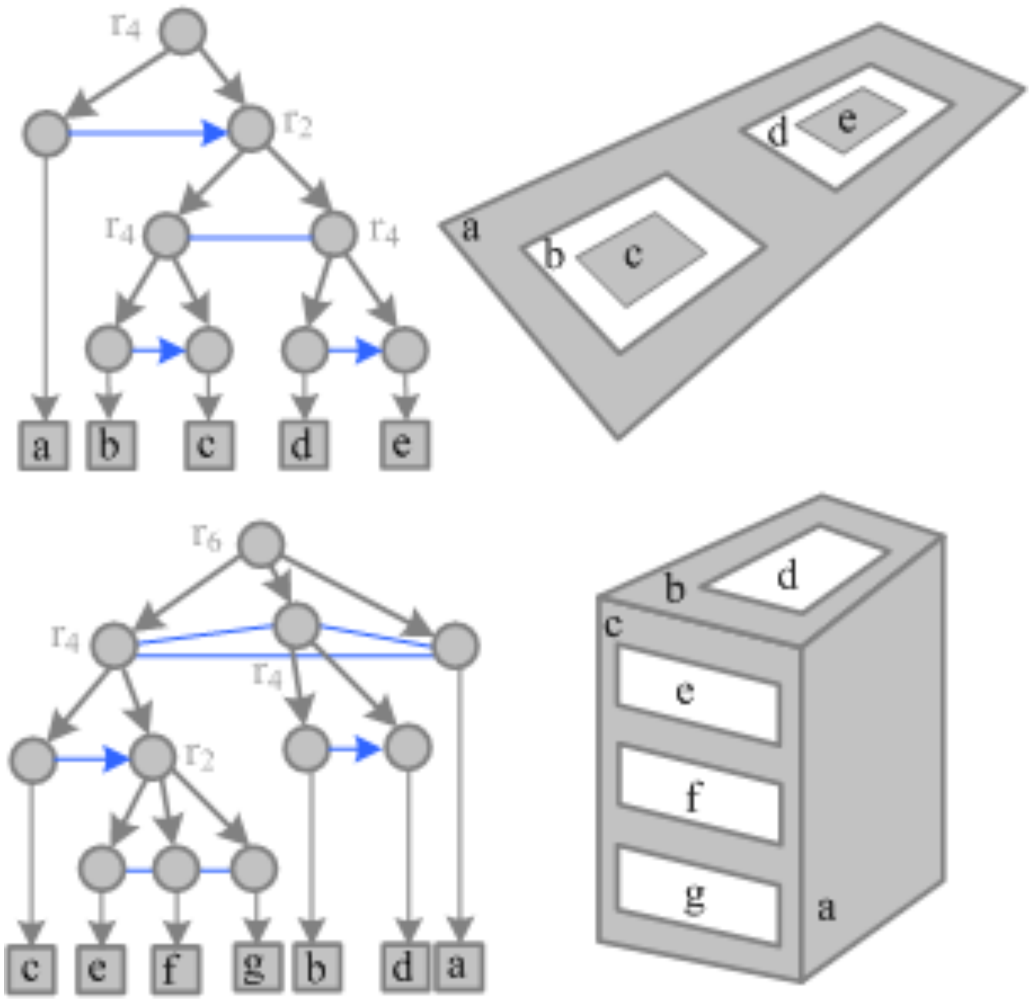


Han and Zhu 2005

Six grammar rules which can be used recursively

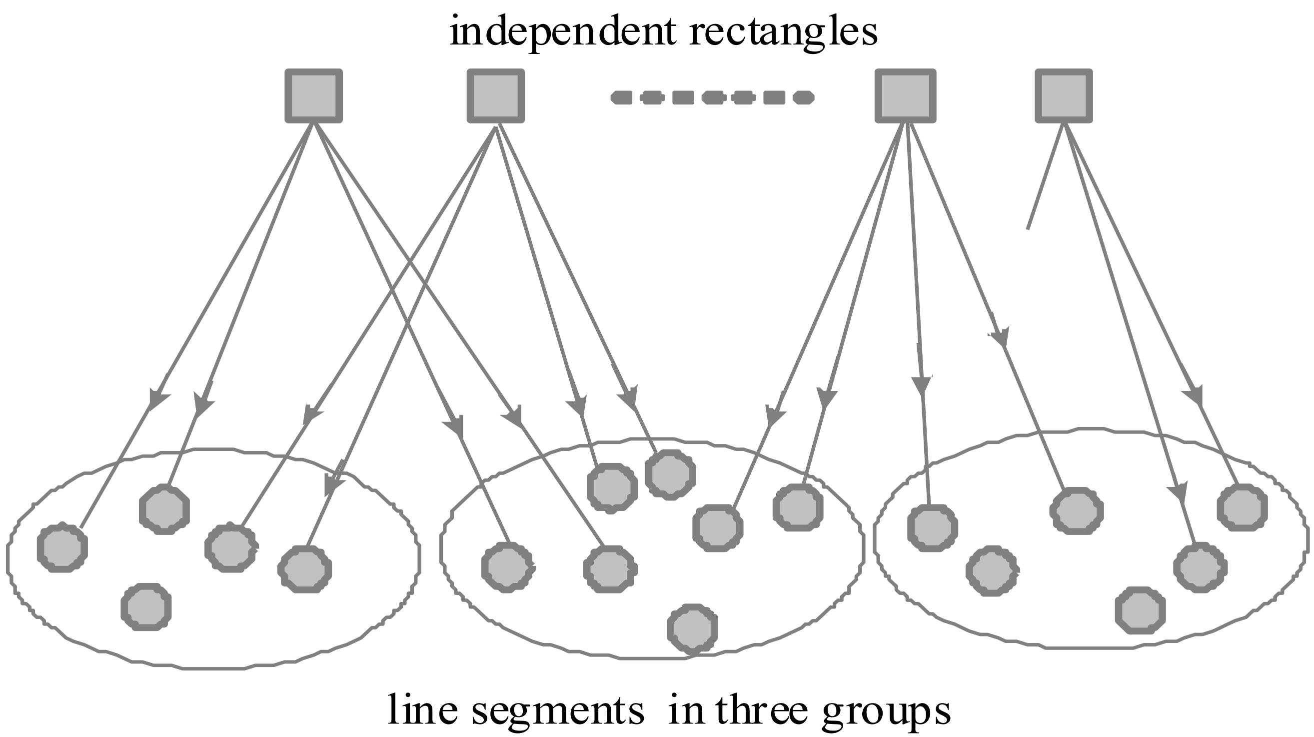


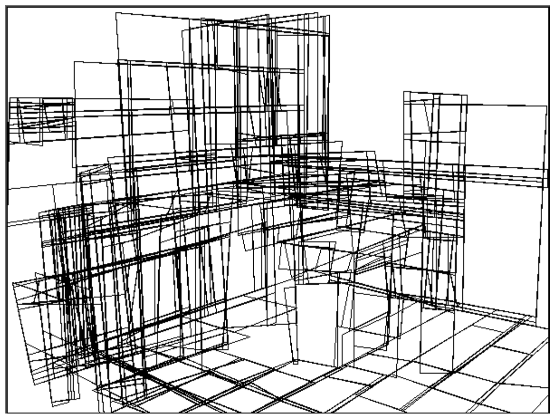
Two configuration examples



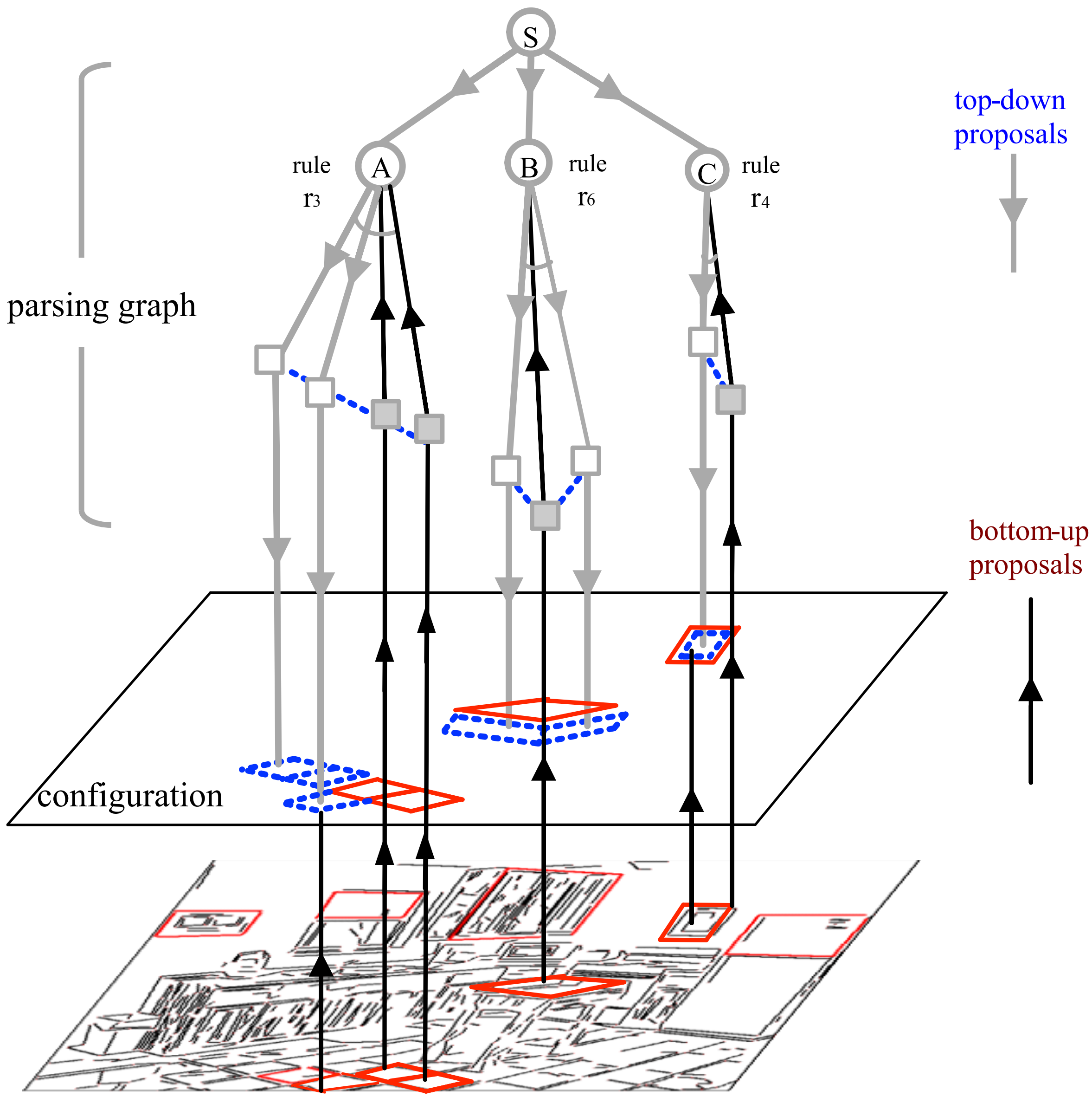
Bottom-up detection (proposal) of rectangles

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



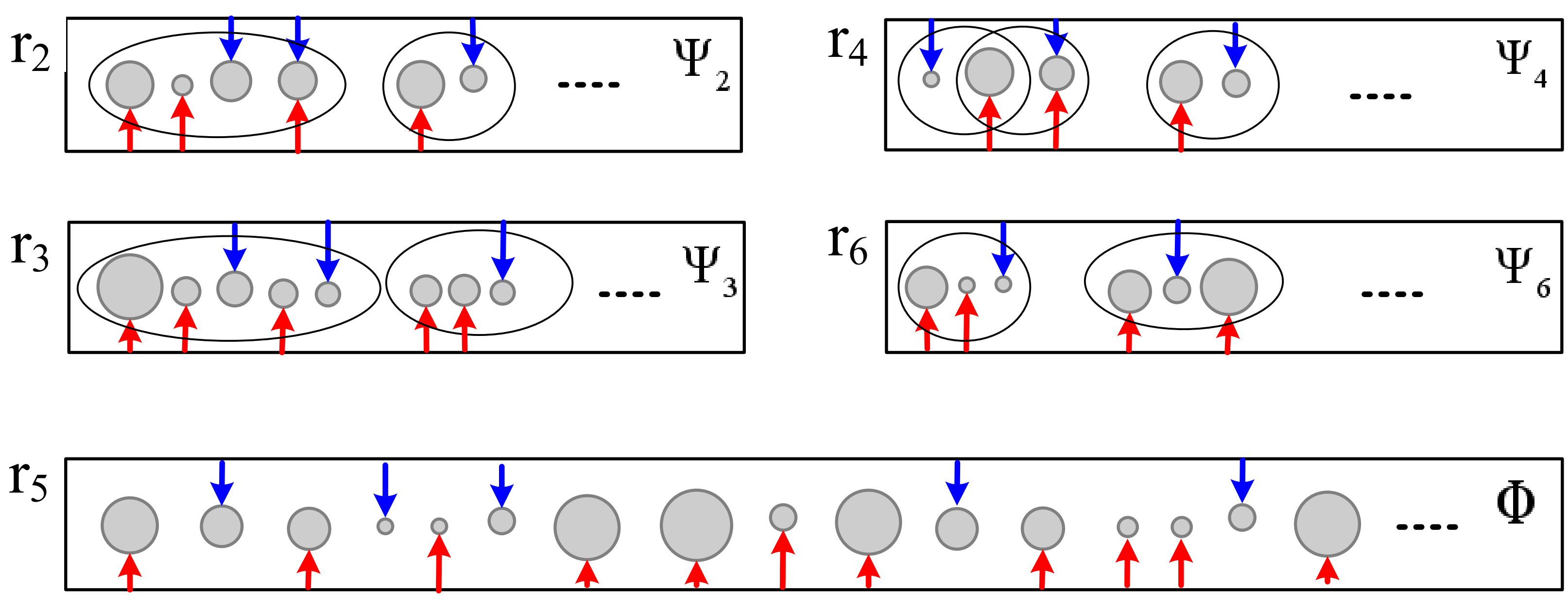


Top-down / bottom-up inference

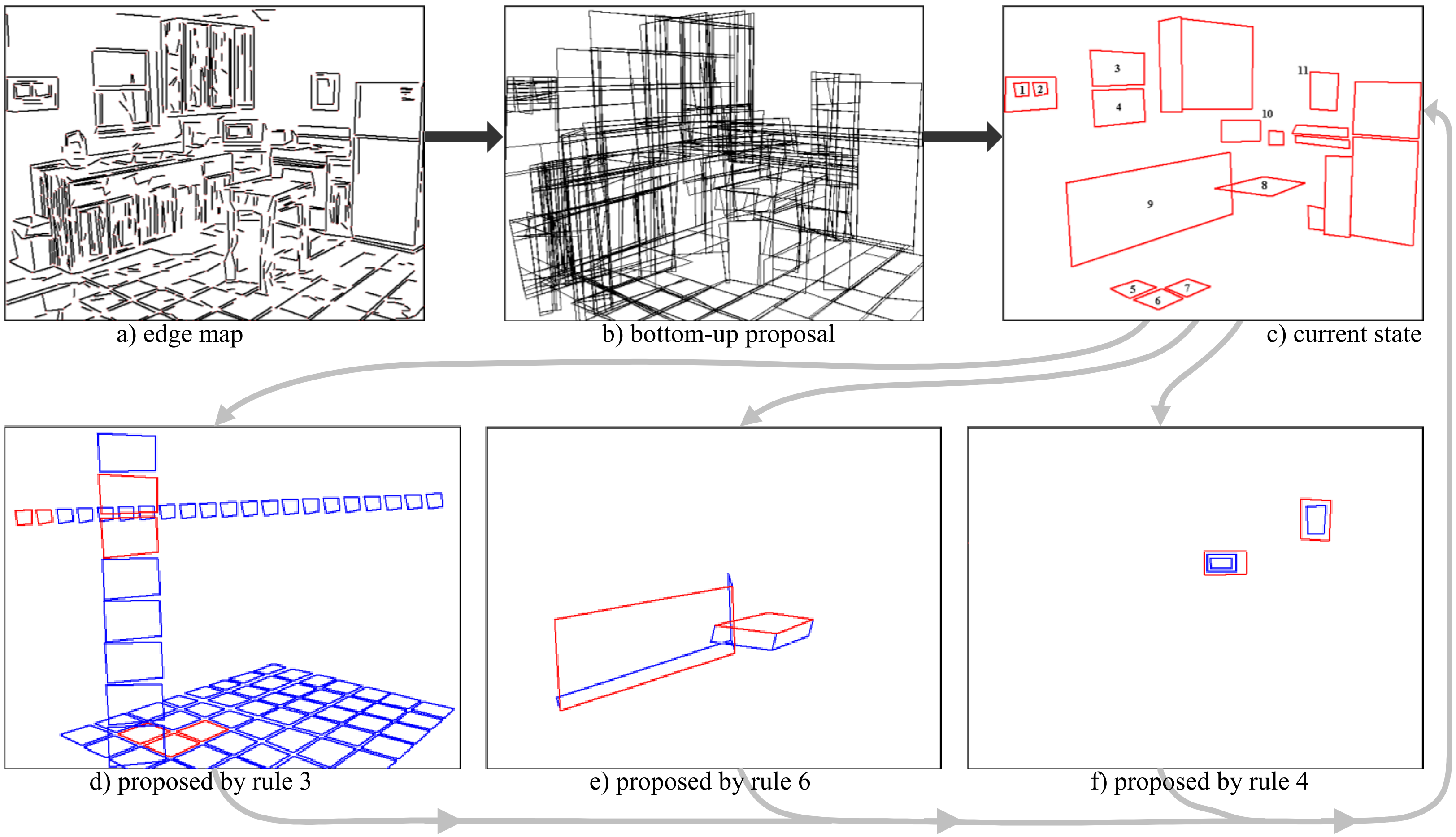


Each grammar rule is an assembly line and maintains an Open-list and Closed-list of particles

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

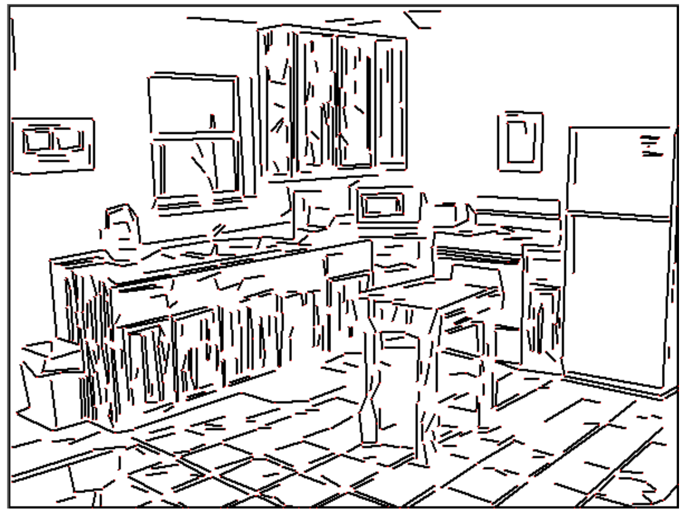


Example of top-down / bottom-up inference

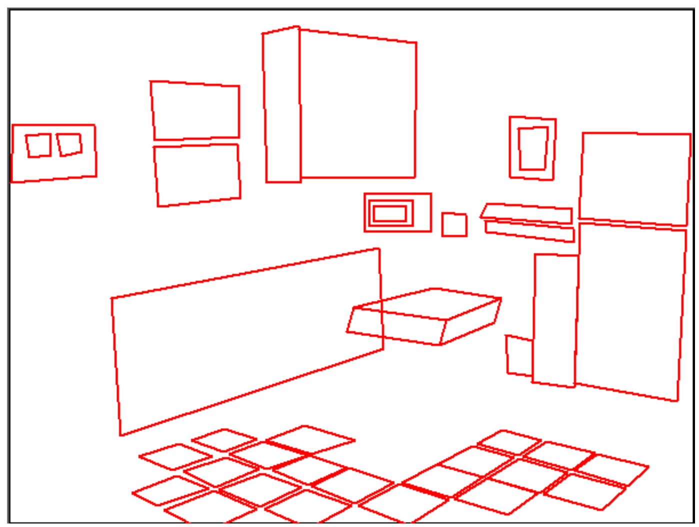


Results

(Han and Zhu, 05)

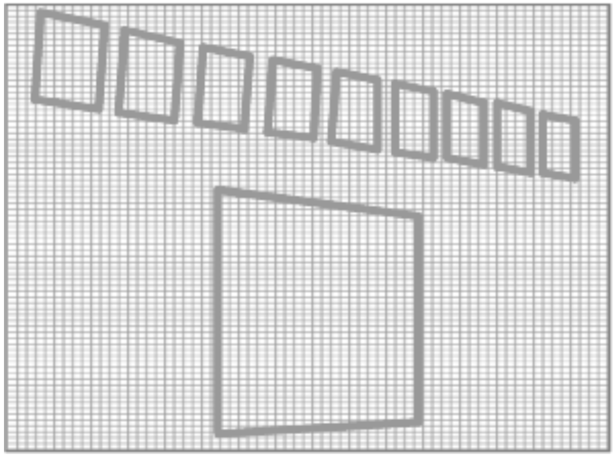


Edge map



Rectangles inferred

Likelihood model based on primal sketch



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

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

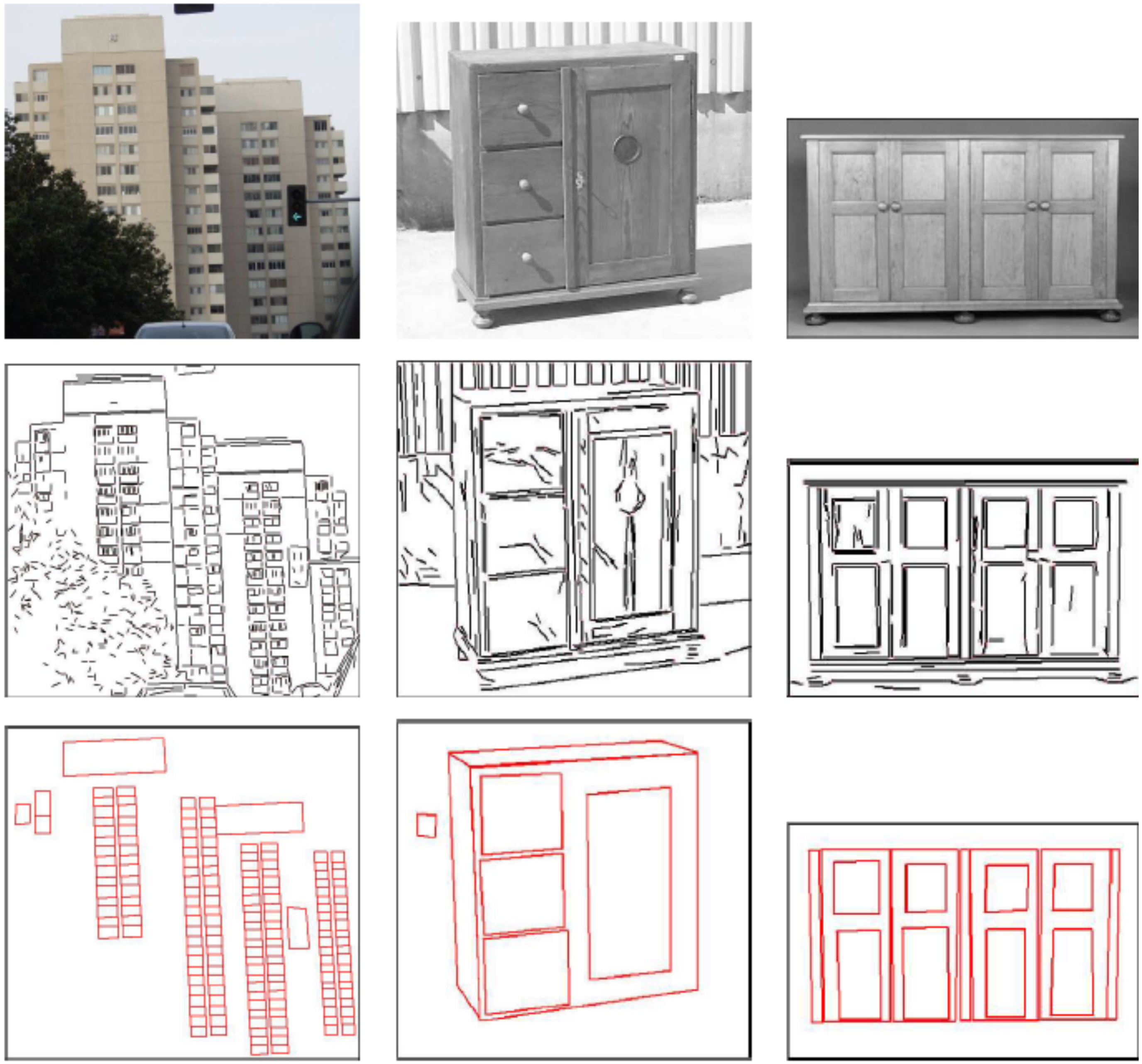
$$\Lambda_{\mathsf{nsk}} = \cup_{m=1}^{M} \Lambda_{\mathsf{nsk,m}},$$

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

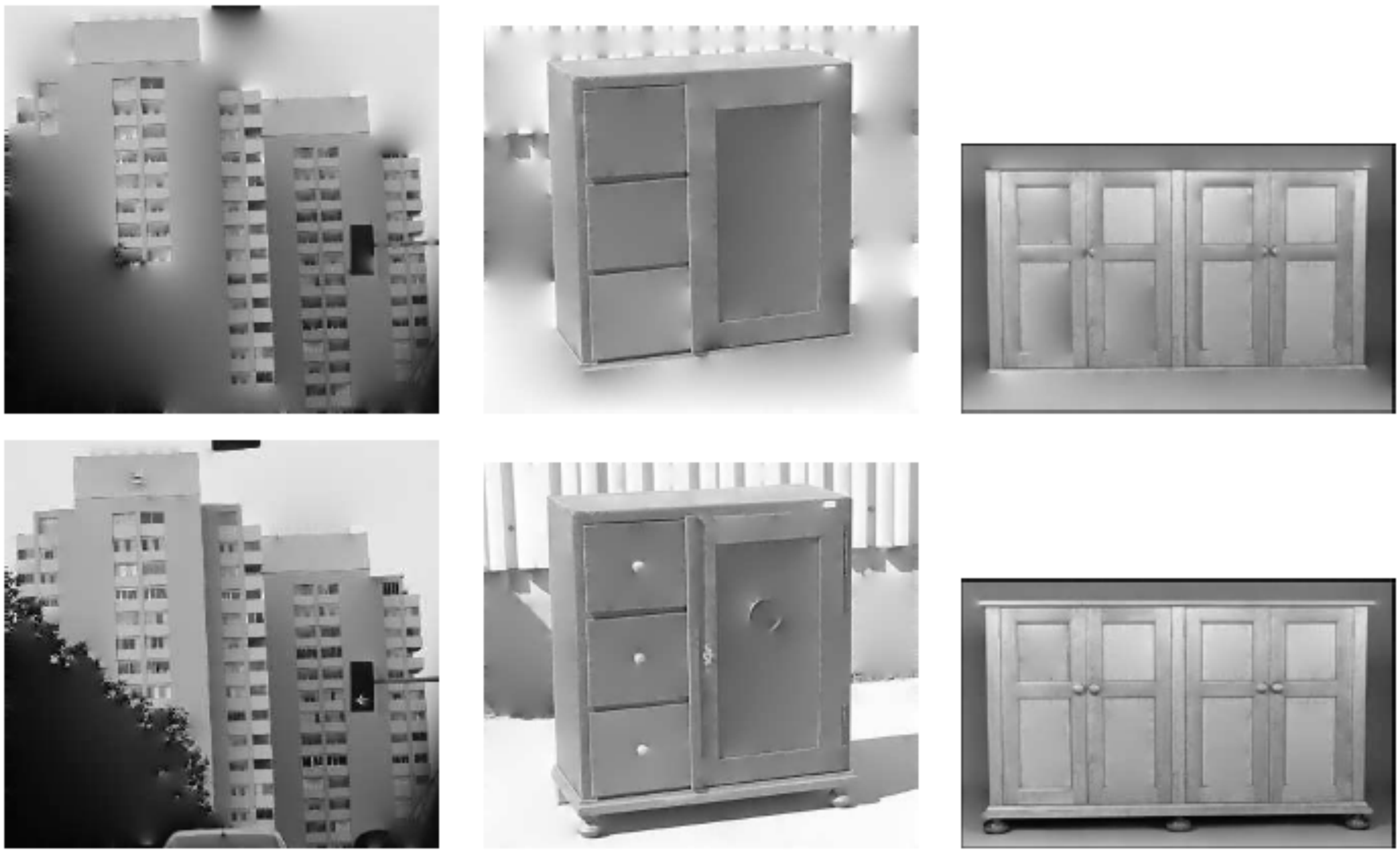
$$p(\mathbf{I}_{\mathsf{sk,k}}|C) \propto \exp\{-\sum_{(x,y)\in\Lambda_{\mathsf{sk,k}}} \frac{(\mathbf{I}(x,y) - B_k(x,y))^2}{2\sigma^2}\}$$

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

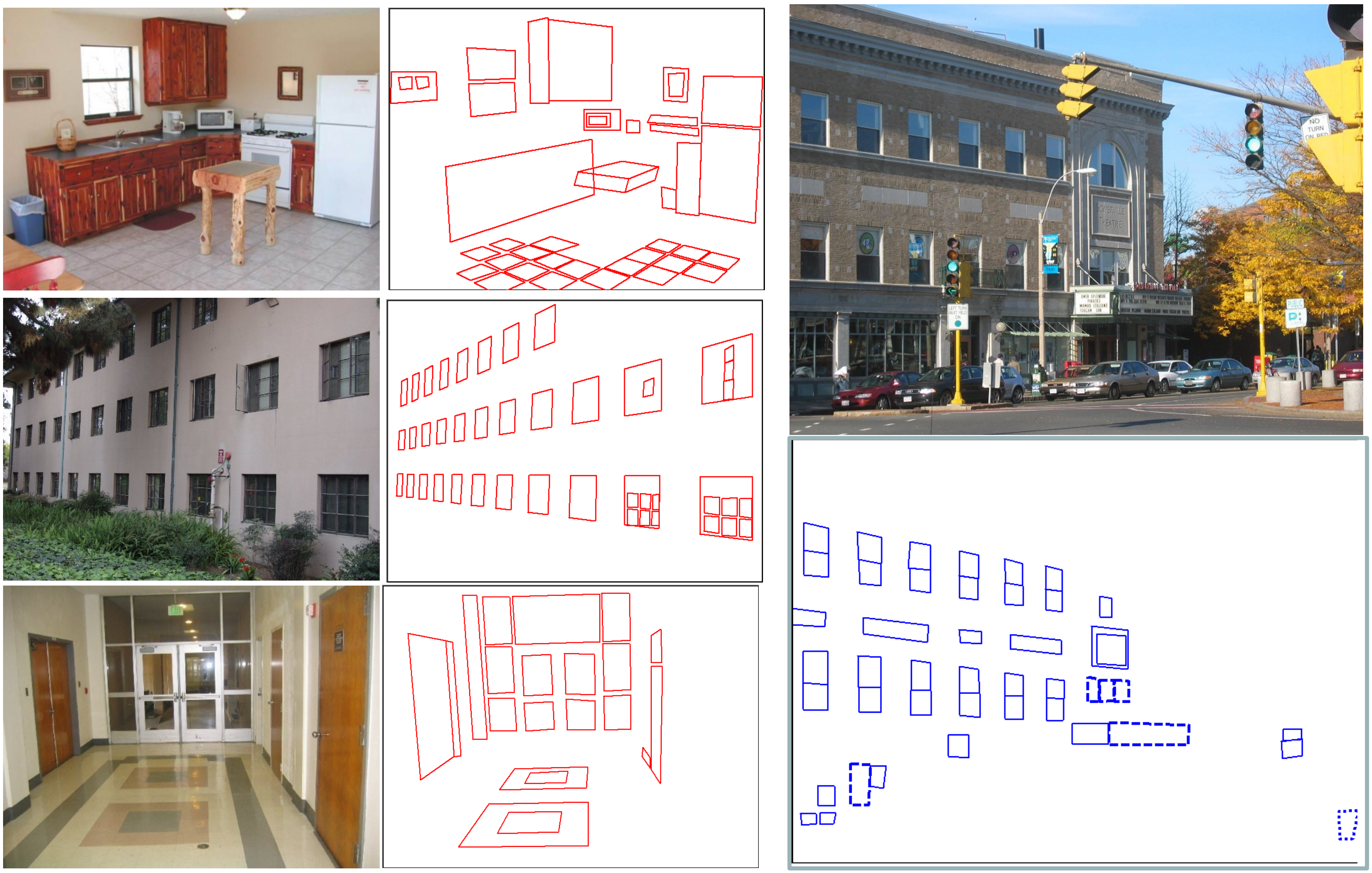
Synthesis based on the parsing model



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