Learning Grammatical Models for 2D Object Recognition

Meg Aycinena

December 8, 2007

Joint work with

Leslie Pack Kaelbling and Tomas Lozano-Perez

Shared Parts and Structure

Shared Parts and Structure

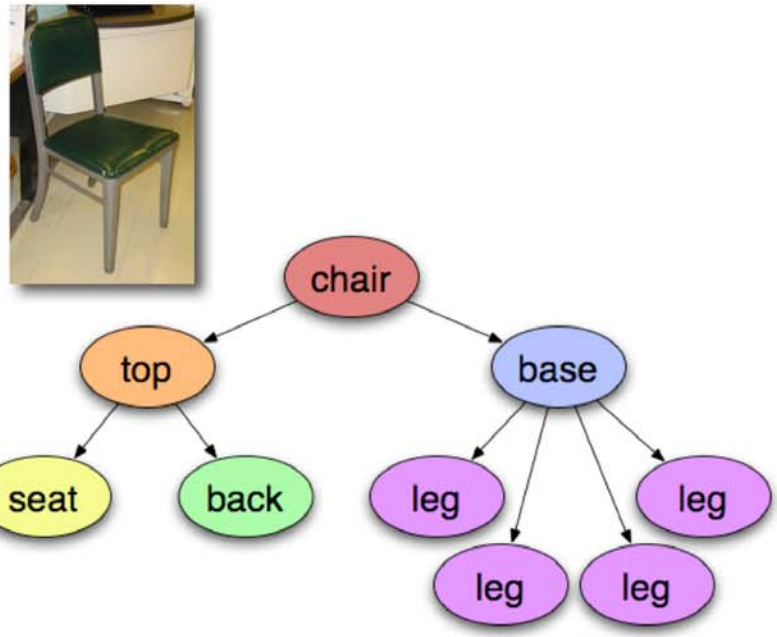
Shared Parts and Structure

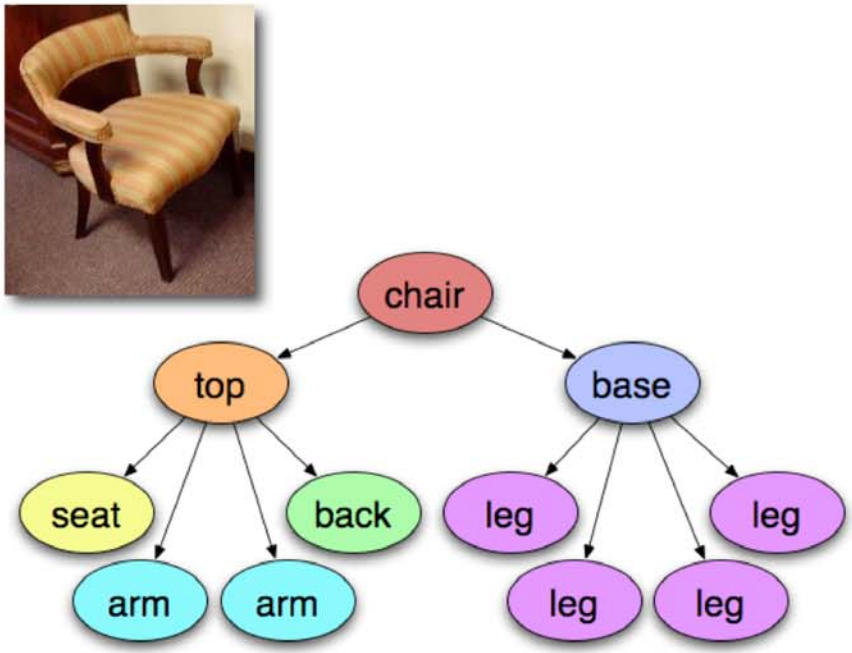
Approach

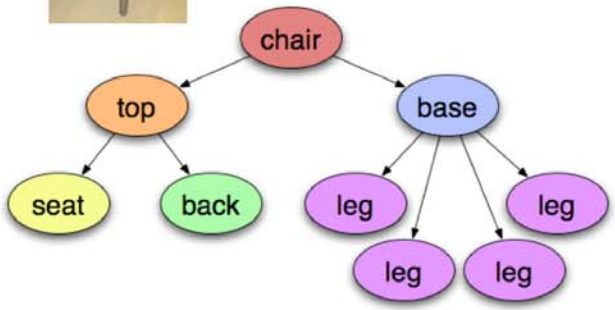
- Capture structural variability in object classes.
- Compactly describe and share common parts and substructure among classes.

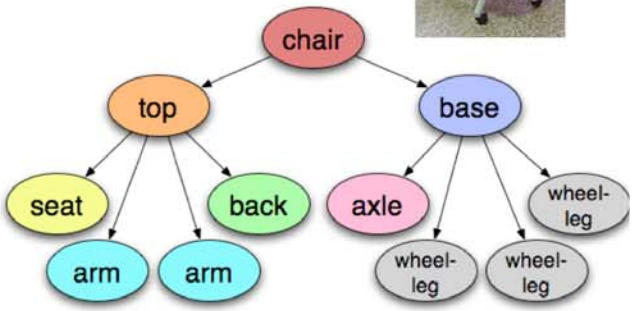
Approach

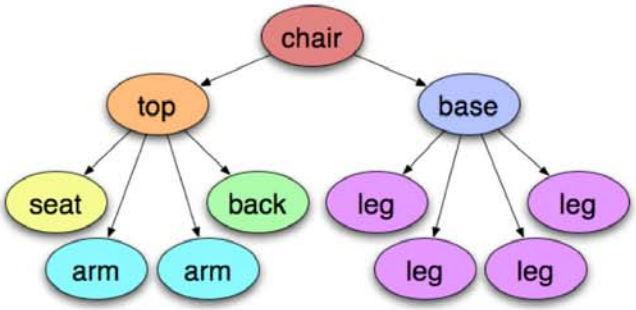
- Capture structural variability in object classes.
- Compactly describe and share common parts and substructure among classes.

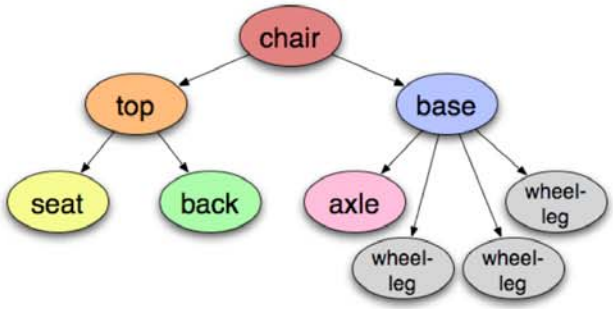




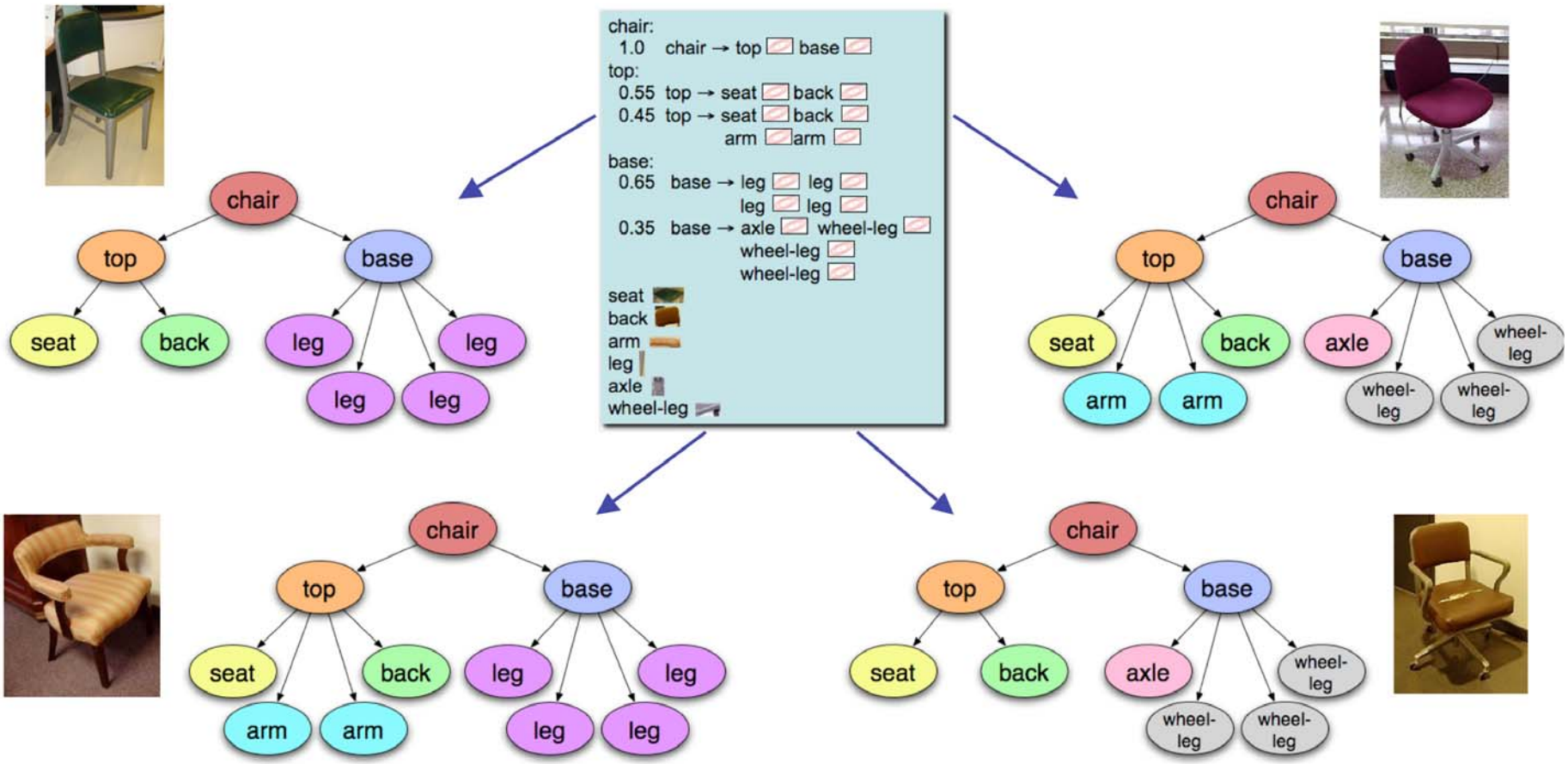

chair
top
base
seat
back
leg
leg
leg
leg


chair
top
base
seat
back
axle
wheel-
leg
arm
arm
wheel-
leg
wheel-
leg


chair
top
base
seat
back
leg
leg
arm
arm
leg
leg


chair
top
base
seat
back
axle
wheel-
leg
wheel-
leg
wheel-
leg

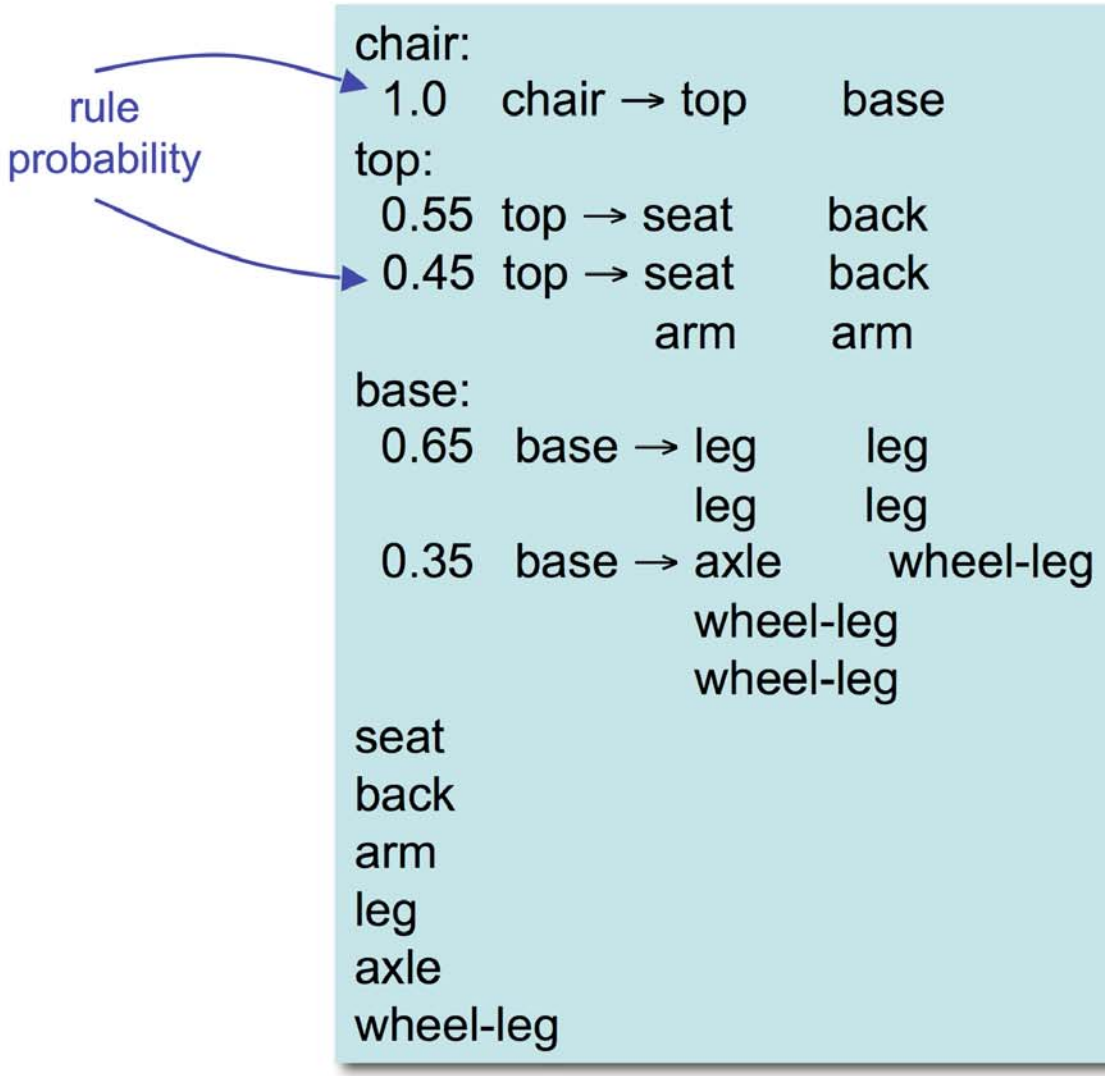
Probabilistic Geometric Grammars (PGGs)



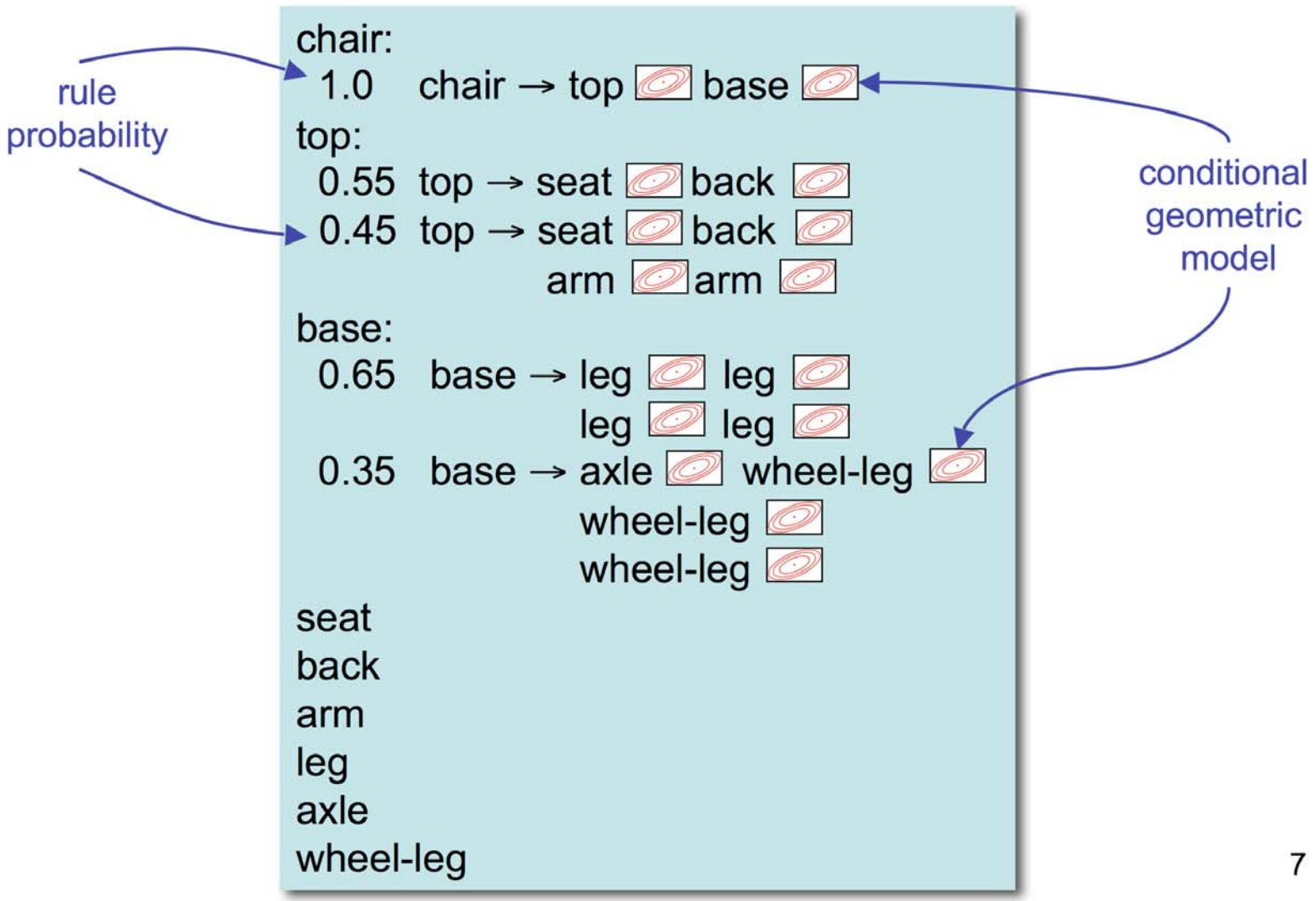
A Probabilistic Geometric Grammar (PGGs)

```
chair:
  1.0   chair → top       base
top:
  0.55  top → seat        back
  0.45  top → seat        back
                arm       arm
base:
  0.65   base → leg         leg
                leg         leg
  0.35   base → axle          wheel-leg
                wheel-leg
                wheel-leg
seat
back
arm
leg
axle
wheel-leg
```

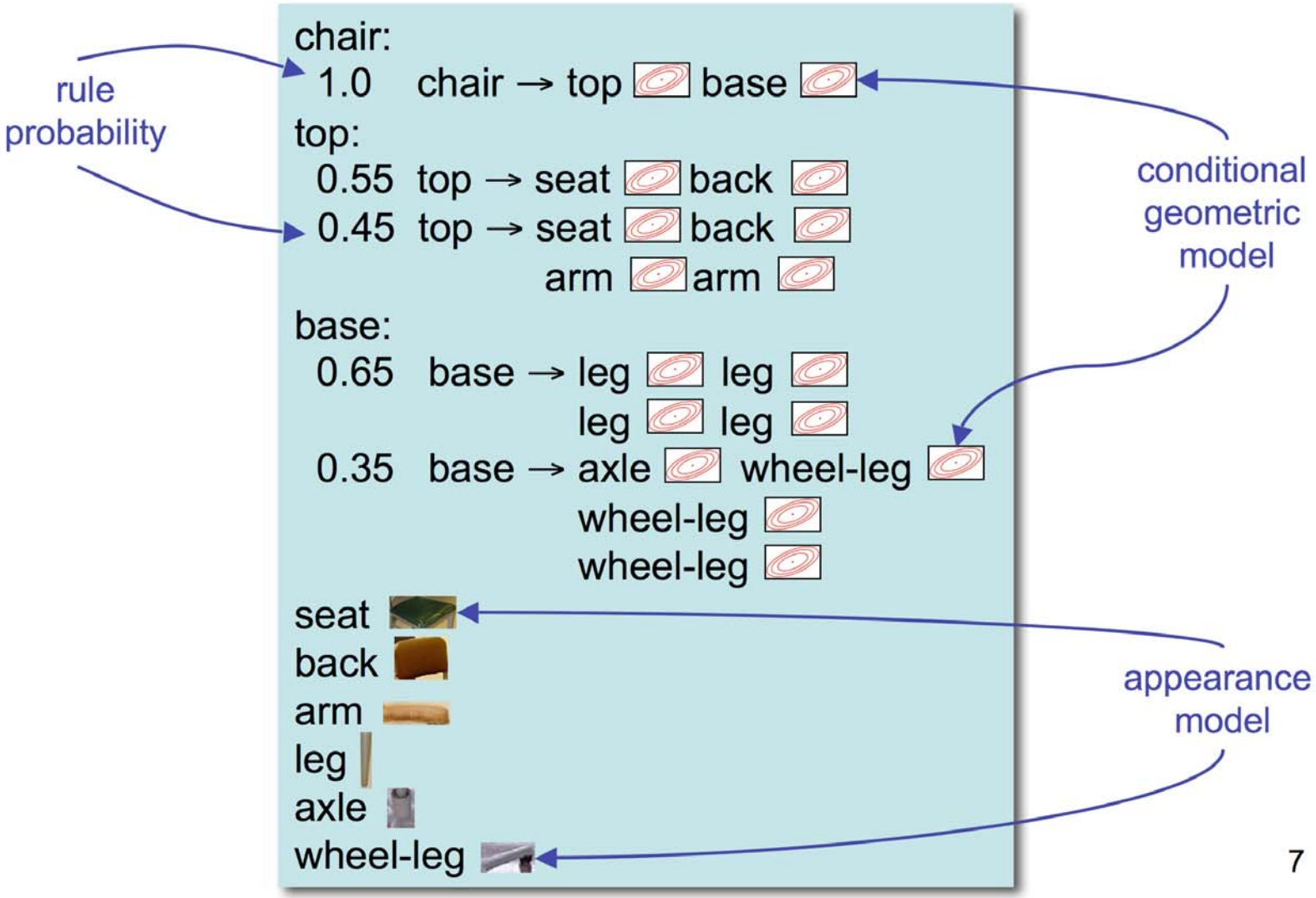
A Probabilistic Geometric Grammar (PGGs)



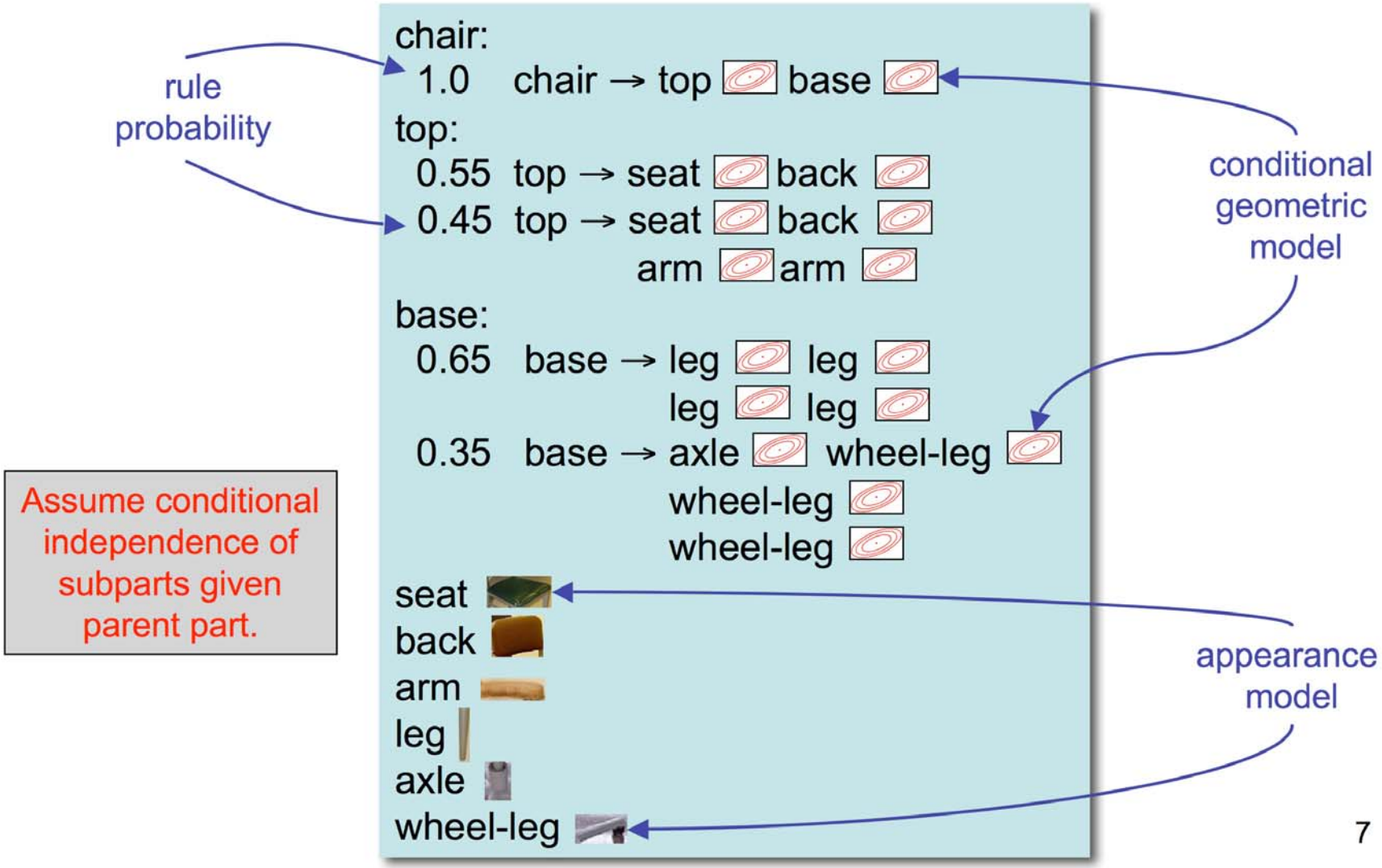
A Probabilistic Geometric Grammar (PGGs)



A Probabilistic Geometric Grammar (PGGs)



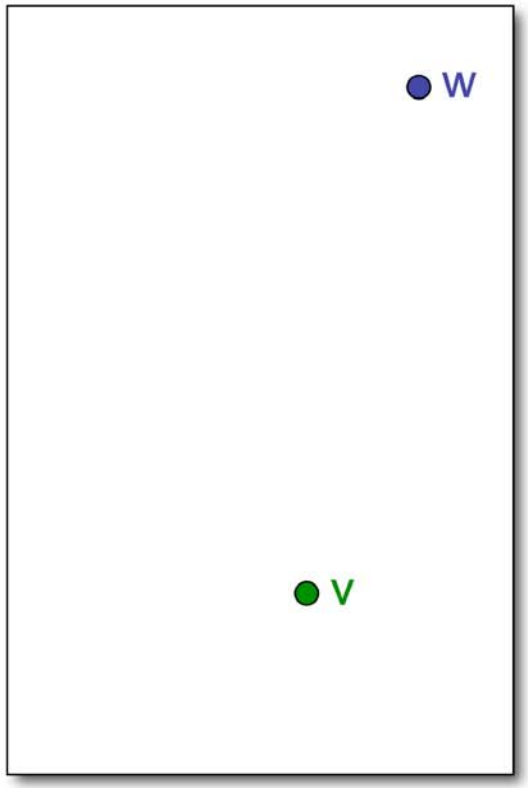
A Probabilistic Geometric Grammar (PGGs)



Conditional Geometry Models

Conditional Geometry Models

- Model any geometric attributes of child part v given parent part w.

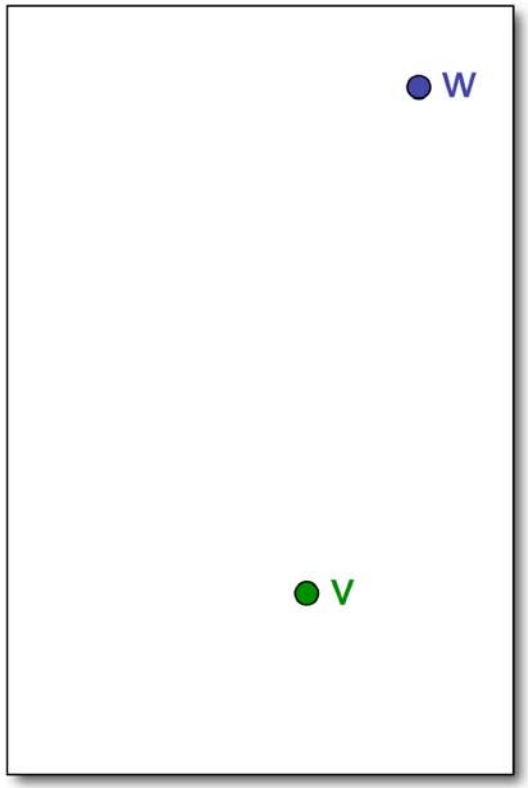


Conditional Geometry Models

- Model any geometric attributes of child part v given parent part w.
- Minimal approach: relative location of centroids.

$$P(v \mid w) = \mathcal{N}(\underbrace{x_v - x_w, y_v - y_w}; \mu, \Sigma)$$

relative location of child centroid to parent

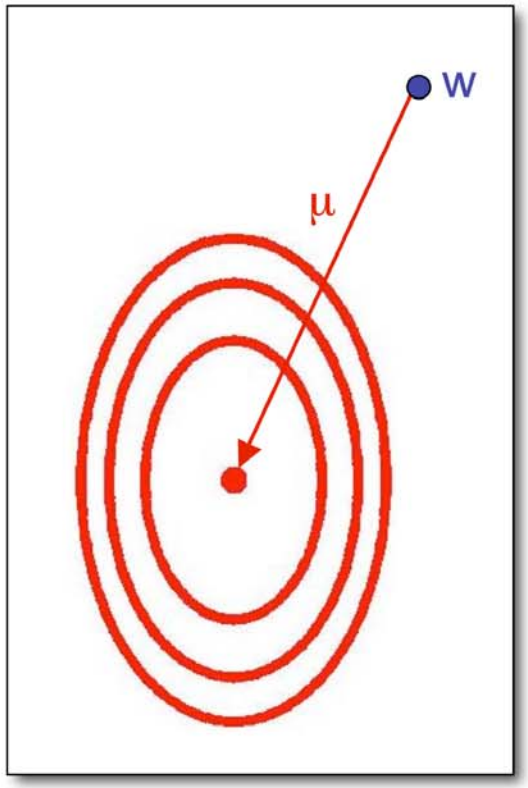


Conditional Geometry Models

- Model any geometric attributes of child part v given parent part w.
- Minimal approach: relative location of centroids.

$$P(v \mid w) = \mathcal{N}(\underbrace{x_v - x_w, y_v - y_w}; \mu, \Sigma)$$

relative location of child centroid to parent

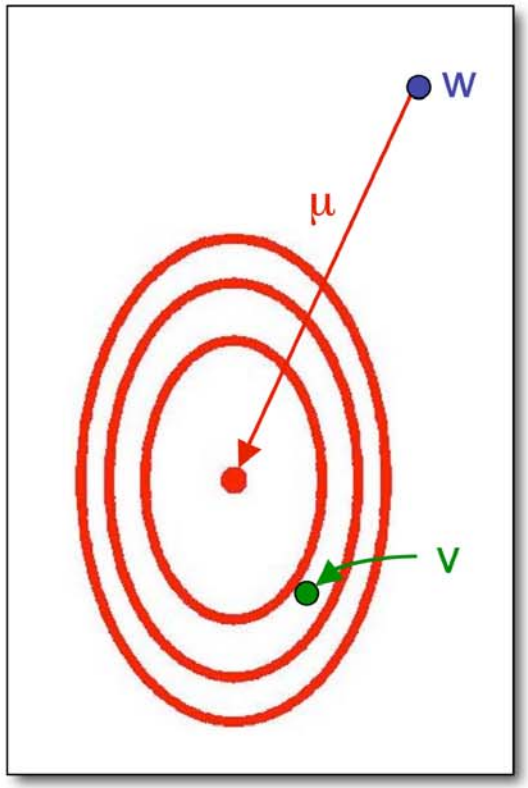


Conditional Geometry Models

- Model any geometric attributes of child part v given parent part w.
- Minimal approach: relative location of centroids.

$$P(v \mid w) = \mathcal{N}(\underbrace{x_v - x_w, y_v - y_w}; \mu, \Sigma)$$

relative location of child centroid to parent



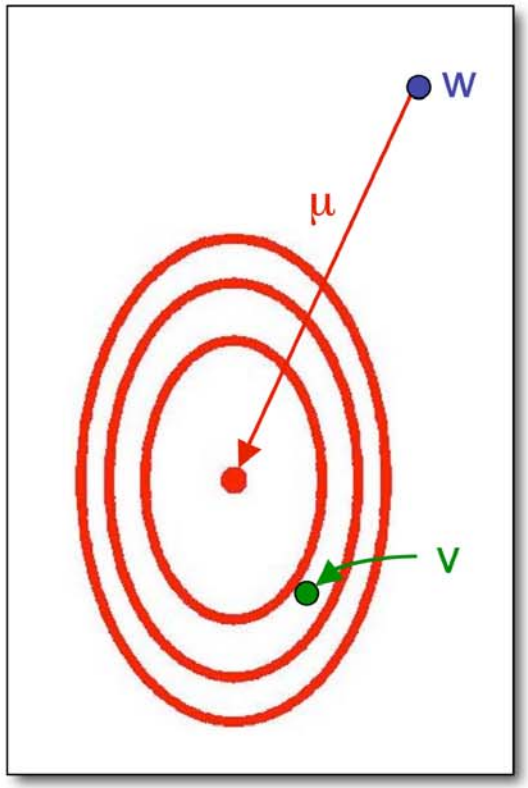
Conditional Geometry Models

- Model any geometric attributes of child part v given parent part w.
- Minimal approach: relative location of centroids.

$$P(v \mid w) = \mathcal{N}(\underbrace{x_v - x_w, y_v - y_w}; \mu, \Sigma)$$

relative location of child centroid to parent

- Many choices of representation!
  - e.g., location, scale, orientation.



Appearance Models

Appearance Models

- Appearance model for each primitive
  part model c must define ratio:

$$\frac{P\left(I_w \mid cw\right)}{P\left(I_w\right)}$$

Appearance Models

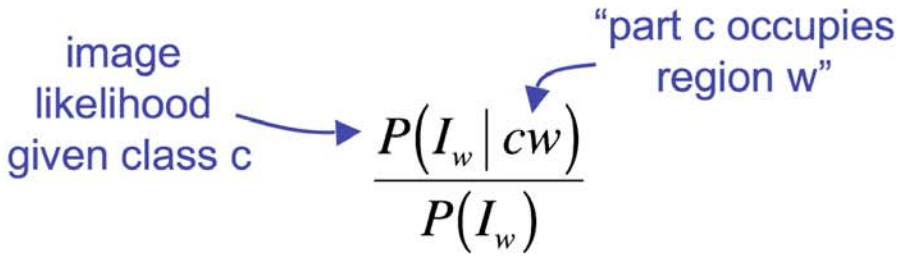
- Appearance model for each primitive part model c must define ratio:

"part c occupies region w"

$$\frac{P(I_w \mid cw)}{P(I_w)}$$

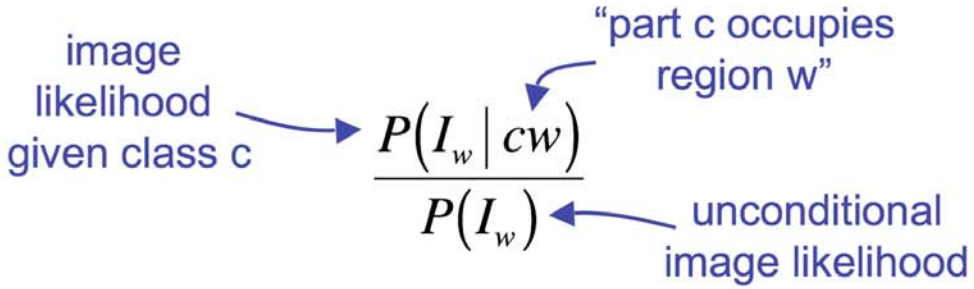
Appearance Models

- Appearance model for each primitive part model c must define ratio:



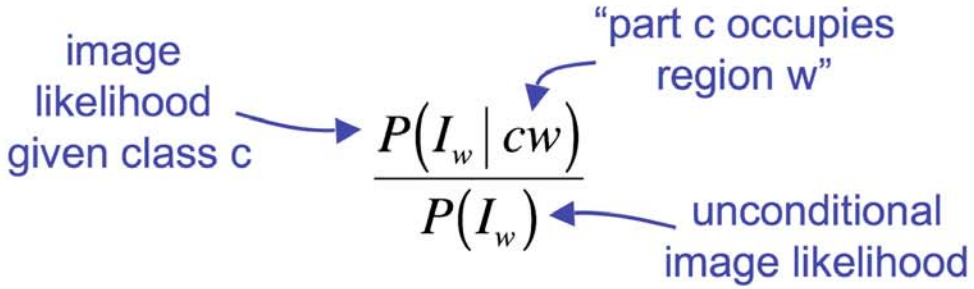
Appearance Models

- Appearance model for each primitive part model c must define ratio:



Appearance Models

- Appearance model for each primitive part model c must define ratio:



- Again, many choices of representation!

Appearance Models

- Appearance model for each primitive part model c must define ratio:

image likelihood given class c

"part c occupies region w"

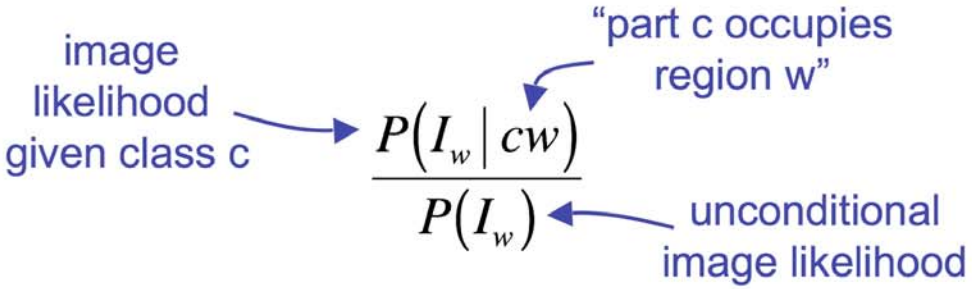
$$\frac{P(I_w \mid cw)}{P(I_w)}$$

unconditional image likelihood

- Again, many choices of representation!
  - intensity or edge templates

Appearance Models

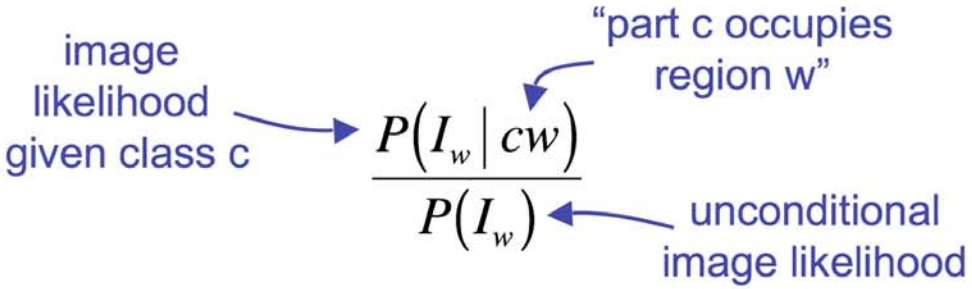
- Appearance model for each primitive part model c must define ratio:



- Again, many choices of representation!
  - intensity or edge templates
  - region shape or curvature

Appearance Models

- Appearance model for each primitive part model c must define ratio:

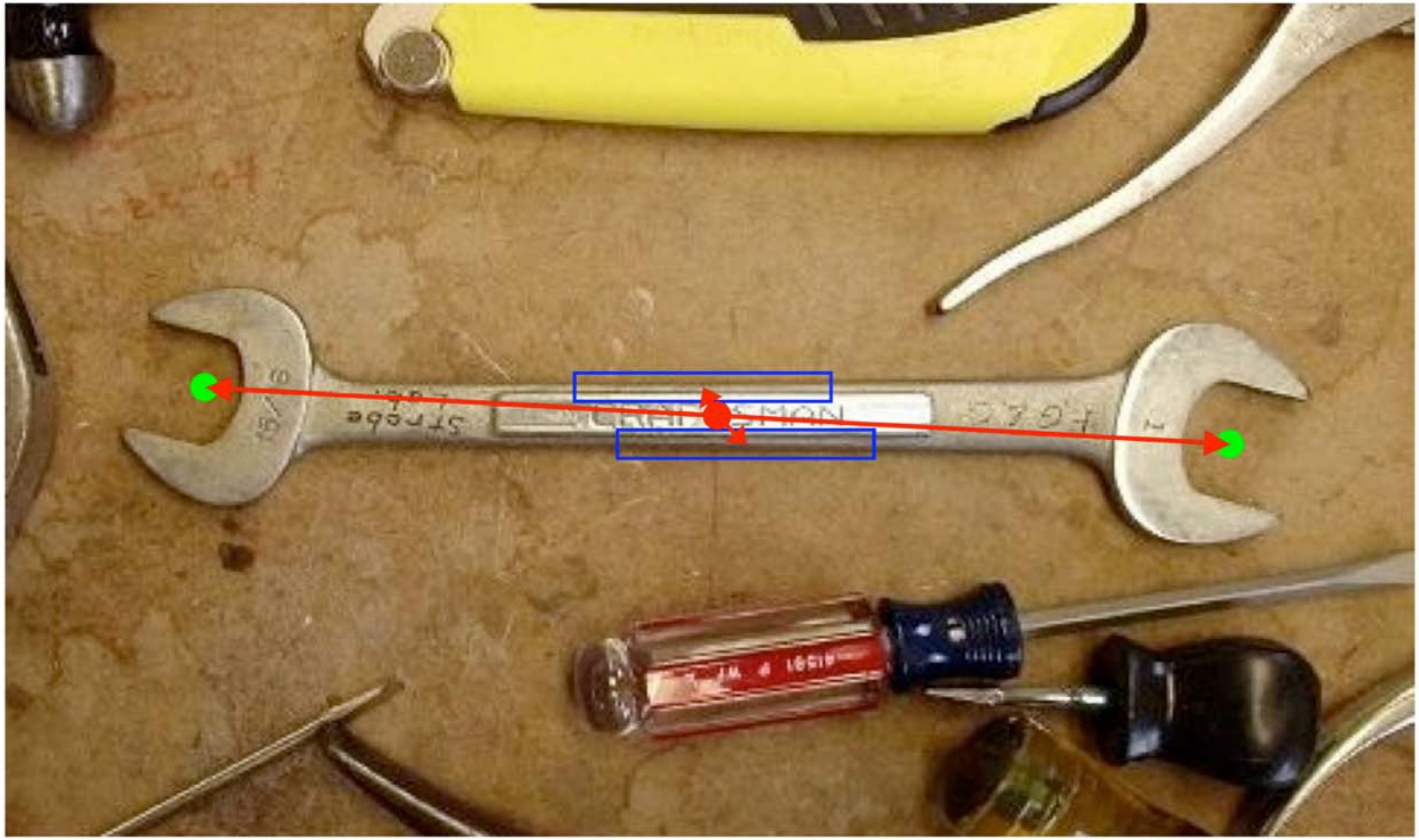


- Again, many choices of representation!
  - intensity or edge templates
  - region shape or curvature
  - distributions over SIFT descriptors of interest points

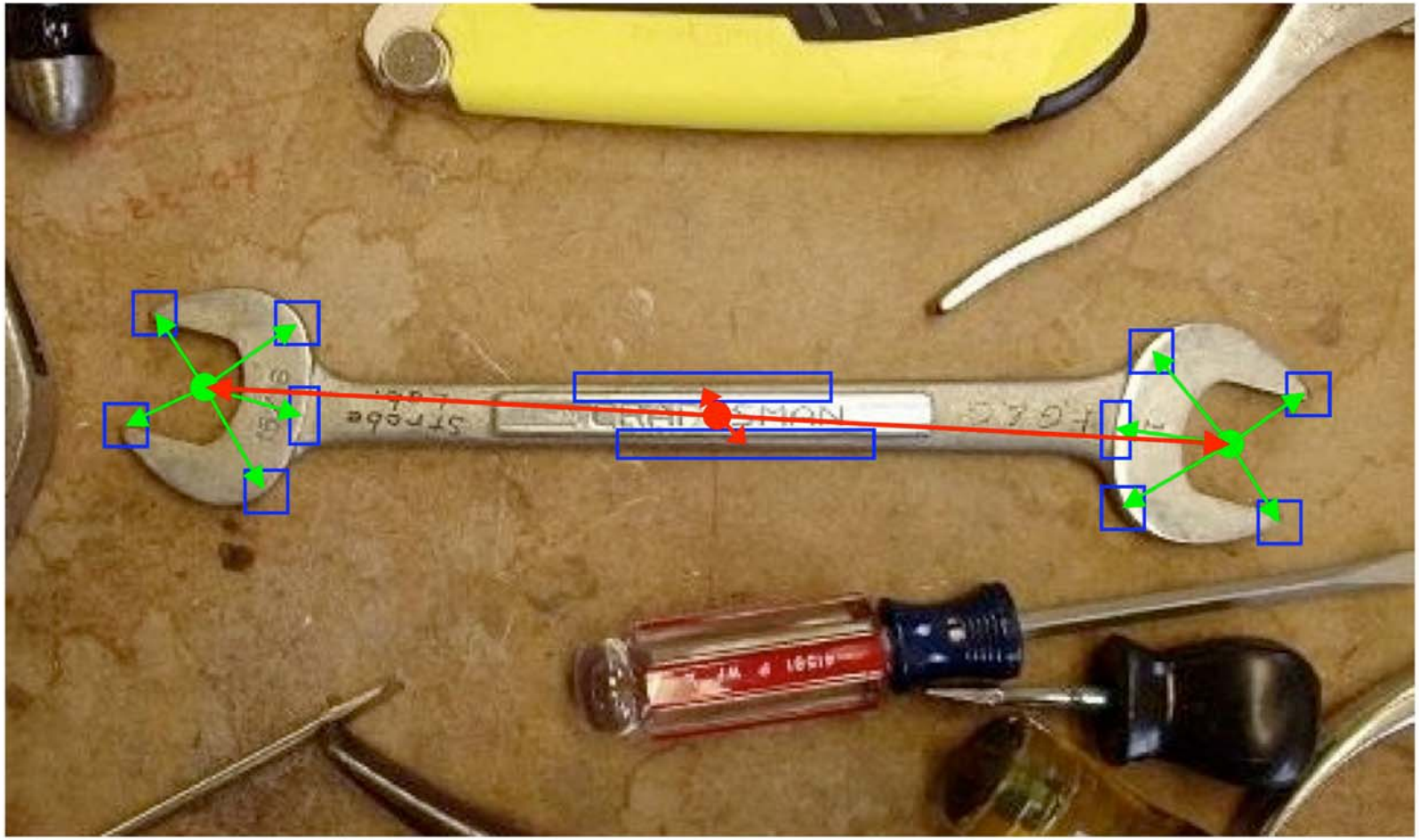
An Example Parse Tree

An Example Parse Tree

An Example Parse Tree



An Example Parse Tree



Efficient Recognition with PGGs

Efficient Recognition with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).

Efficient Recognition with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Recognition with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.
- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.

Efficient Recognition with PGGs

Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \underbrace{\frac{P(I_w \mid cw)}{P(I_w)}}_{\text{image likelihood ratio}}$$

"part c occupies region w"

Efficient Recognition with PGGs

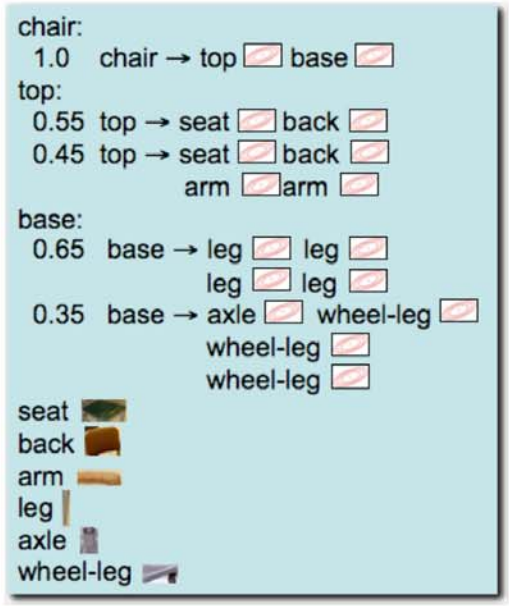
- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio

- Recursively decompose image likelihood ratio according to grammar:



Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

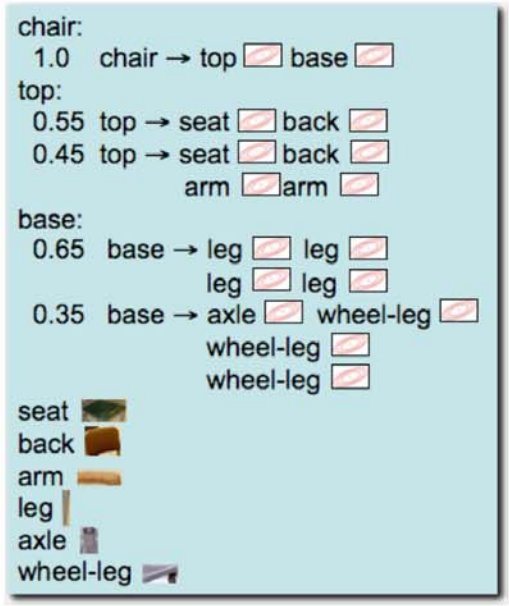
$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio

- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$



Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

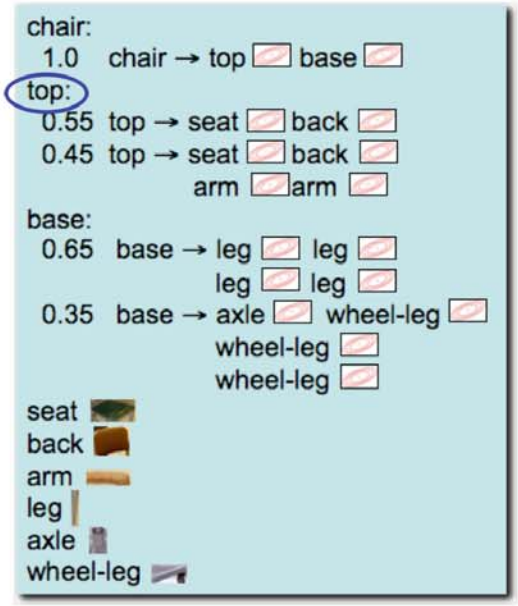
$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio

- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$



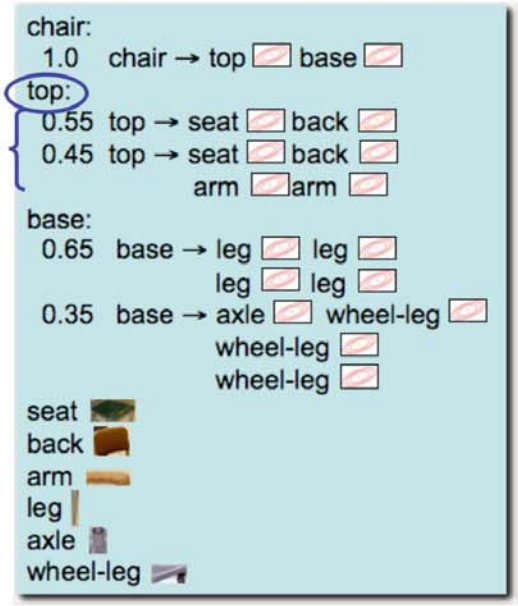
Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio



- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

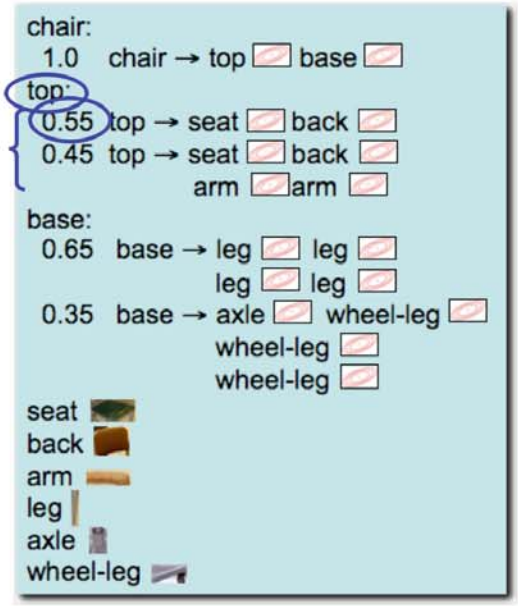
image likelihood ratio

- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

rule probability



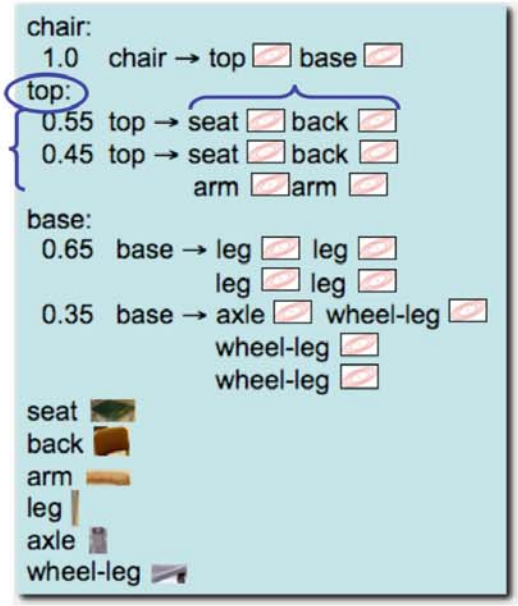
Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio



- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

rule probability

product over rules parts

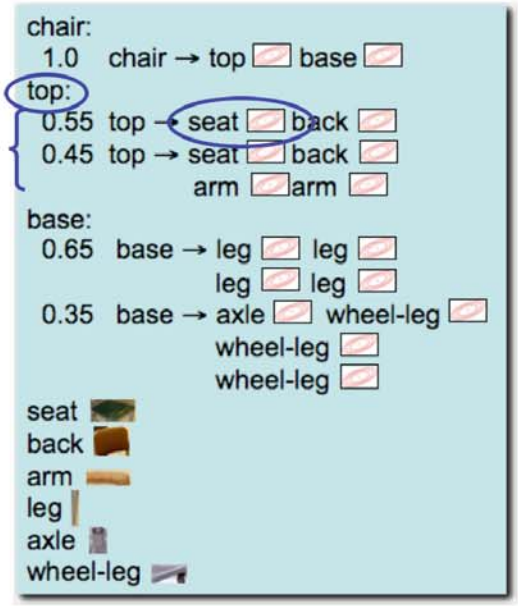
Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio



- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

rule probability

product over rules parts

sum over child regions

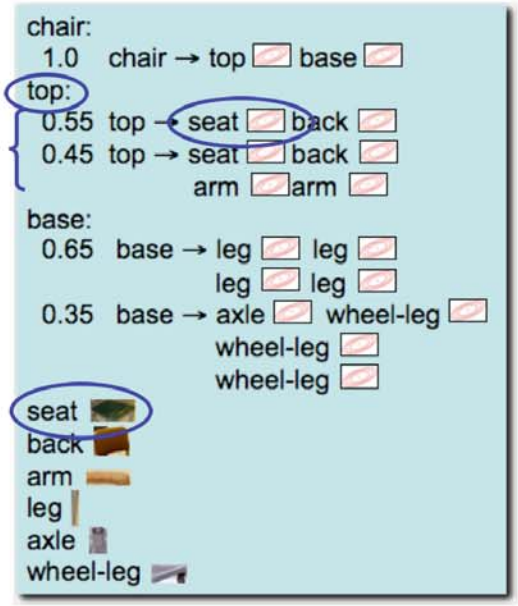
Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio



- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

rule probability

recursive image likelihood ratio

product over rules parts

sum over child regions

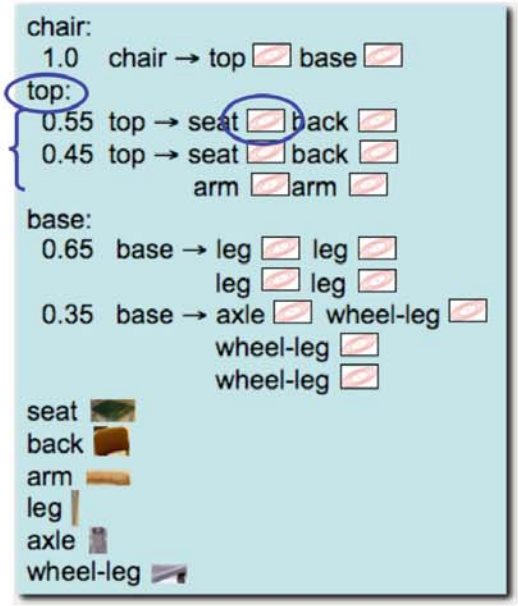
Efficient Recognition with PGGs

- Find most likely region *w** in image *I* with root class *c**:

$$cw^* = \arg\max_{cw} \frac{P(I_w \mid cw)}{P(I_w)}$$

"part c occupies region w"

image likelihood ratio



- Recursively decompose image likelihood ratio according to grammar:

$$\frac{P(I_w \mid cw)}{P(I_w)} = \sum_r P(r \mid c) \prod_k \sum_v \frac{P(I_v \mid d_{crk} v)}{P(I_v)} P(v \mid w; \phi_{crk})$$

sum over rules

rule probability

product over rules parts

sum over child regions

recursive image likelihood ratio

conditional geometric model

Efficient Recognition with PGGs

Efficient Recognition with PGGs

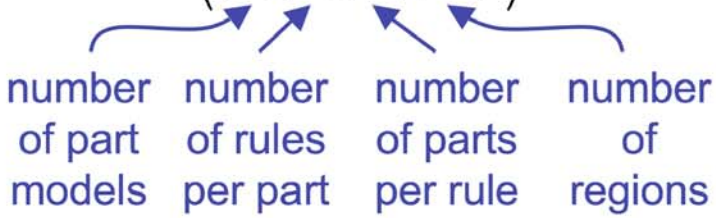
- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.

Efficient Recognition with PGGs

- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.
- We must sum rather than max over tree structures and locations to account for variable number of parts!

Efficient Recognition with PGGs

- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.
- We must sum rather than max over tree structures and locations to account for variable number of parts!
- Complexity: $O\left(|C||R||K||W|^2\right)$



Efficient Recognition with PGGs

- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.
- We must sum rather than max over tree structures and locations to account for variable number of parts!
- Complexity: $O\left(|C||R||K||W|^2\right)$

number of part models — number of rules per part — number of parts per rule — number of regions

Only sum over child regions close to expected position!

Efficient Recognition with PGGs

- Dynamic programming algorithm: recursively calculates score for each region and part model, caching intermediate results.
- We must sum rather than max over tree structures and locations to account for variable number of parts!
- Complexity: $O\left(|C||R||K||W|^2\right)$

number of part models — number of rules per part — number of parts per rule — number of regions

Only sum over child regions close to expected position!

- **Keys to success:**
  - Discrete set of image regions.
  - Conditional independence among child parts.
  - Search constrained by geometry models.

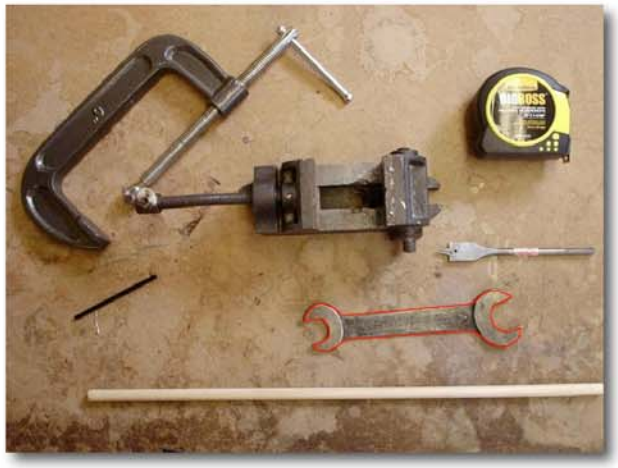
Learning PGG Models

Learning PGG Models

- Goal: Find compact grammar structure and parameters that explain the training data.

Learning PGG Models

- Goal: Find compact grammar structure and parameters that explain the training data.
- Training data: Images labeled with object class and silhouette.



Learning PGG Models

- Goal: Find compact grammar structure and parameters that explain the training data.
- Training data: Images labeled with object class and silhouette.

Learning PGG Models

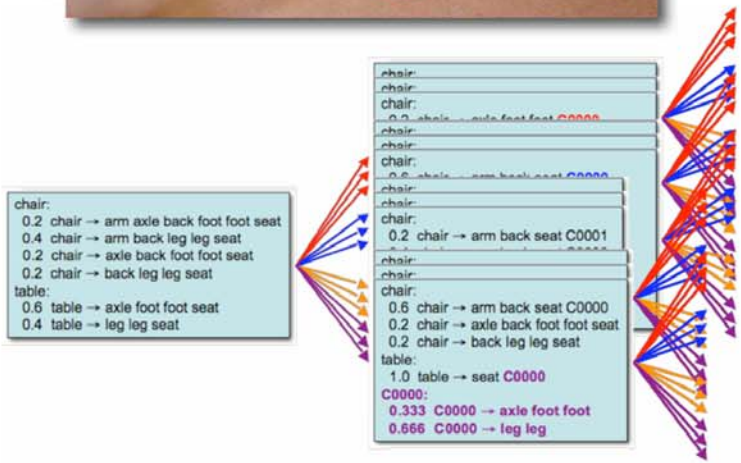
- Goal: Find compact grammar structure and parameters that explain the training data.
- Training data: Images labeled with object class and silhouette.

Learning PGG Models

- Goal: Find compact grammar structure and parameters that explain the training data.
- Training data: Images labeled with object class and silhouette.
- Learn:
  - **Parameters:** estimate from data using EM.

Learning PGG Models

- Goal: Find compact grammar structure and parameters that explain the training data.
- Training data: Images labeled with object class and silhouette.
- Learn:
  - **Parameters:** estimate from data using EM.
  - **Structure:** incremental search through space of grammars.



Parameter Learning in PGGs

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.
- **Parameters:**

γ_{cr} : probability for rule r in part c

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.
- **Parameters:**

γ_{cr} : probability for rule r in part c

μ_{crk}, Σ_{crk} : mean, covariance for rule part k in rule r in part c

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.
- **Parameters:**

γ_{cr} : probability for rule r in part c

μ_{crk}, Σ_{crk} : mean, covariance for rule part k in rule r in part c

- **EM algorithm** extends inside-outside algorithm for PCFGs:

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.
- **Parameters:**

γ_{cr} : probability for rule r in part c

μ_{crk}, Σ_{crk} : mean, covariance for rule part k in rule r in part c

- **EM algorithm** extends inside-outside algorithm for PCFGs:
  - E-step: Calculate responsibilities for each grammar component in each region of each training image.

Parameter Learning in PGGs

- Assume grammar structure is fixed and specified.
- Assume labeled object bounding boxes with class labels.
- **Parameters:**

γ_{cr} : probability for rule r in part c

μ_{crk}, Σ_{crk} : mean, covariance for rule part k in rule r in part c

- **EM algorithm** extends inside-outside algorithm for PCFGs:
  - E-step: Calculate responsibilities for each grammar component in each region of each training image.
  - M-step: Reestimate parameters from data, weighted by responsibilities.

Structure Learning in PGGs

Structure Learning in PGGs

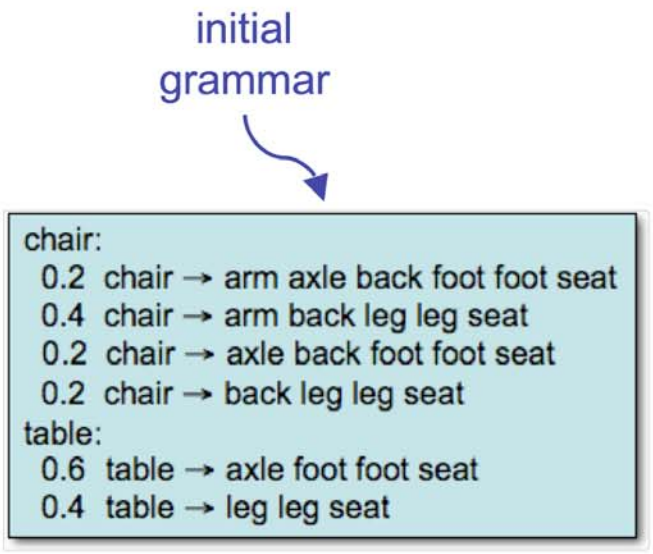
- **Goal:** Find compact grammar that fits training data well.

Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.

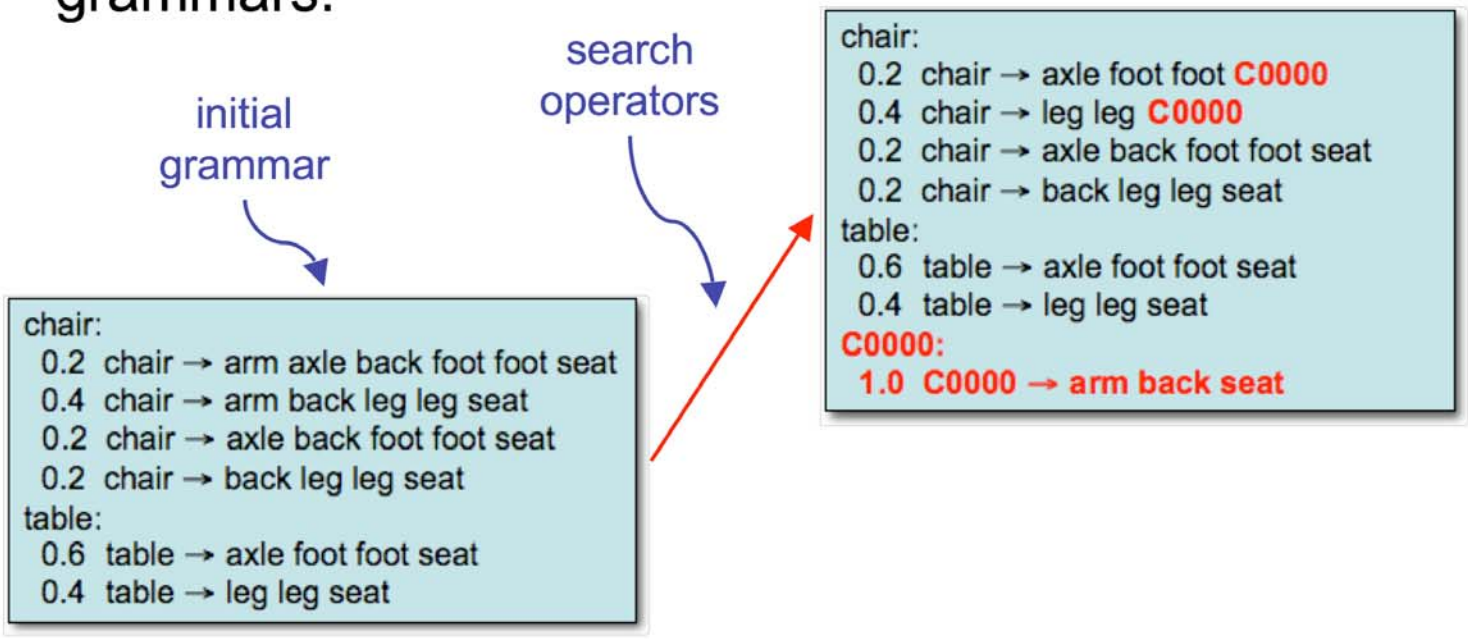
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



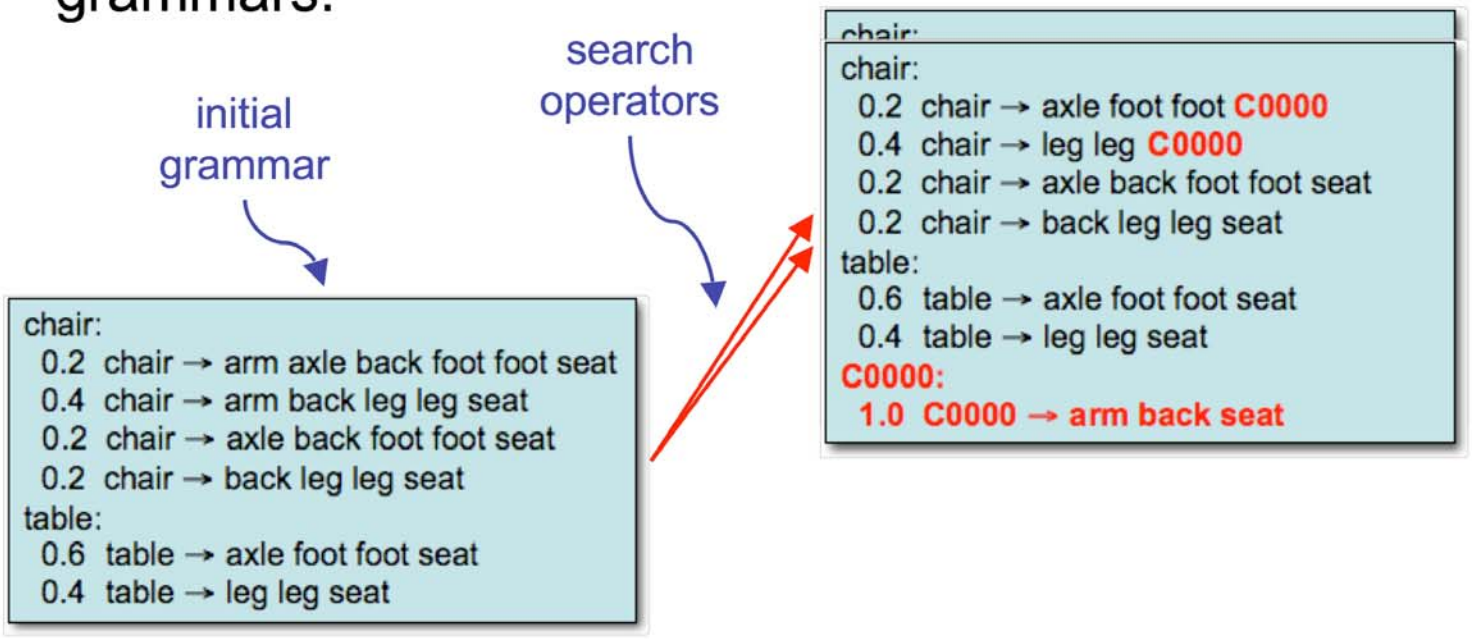
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



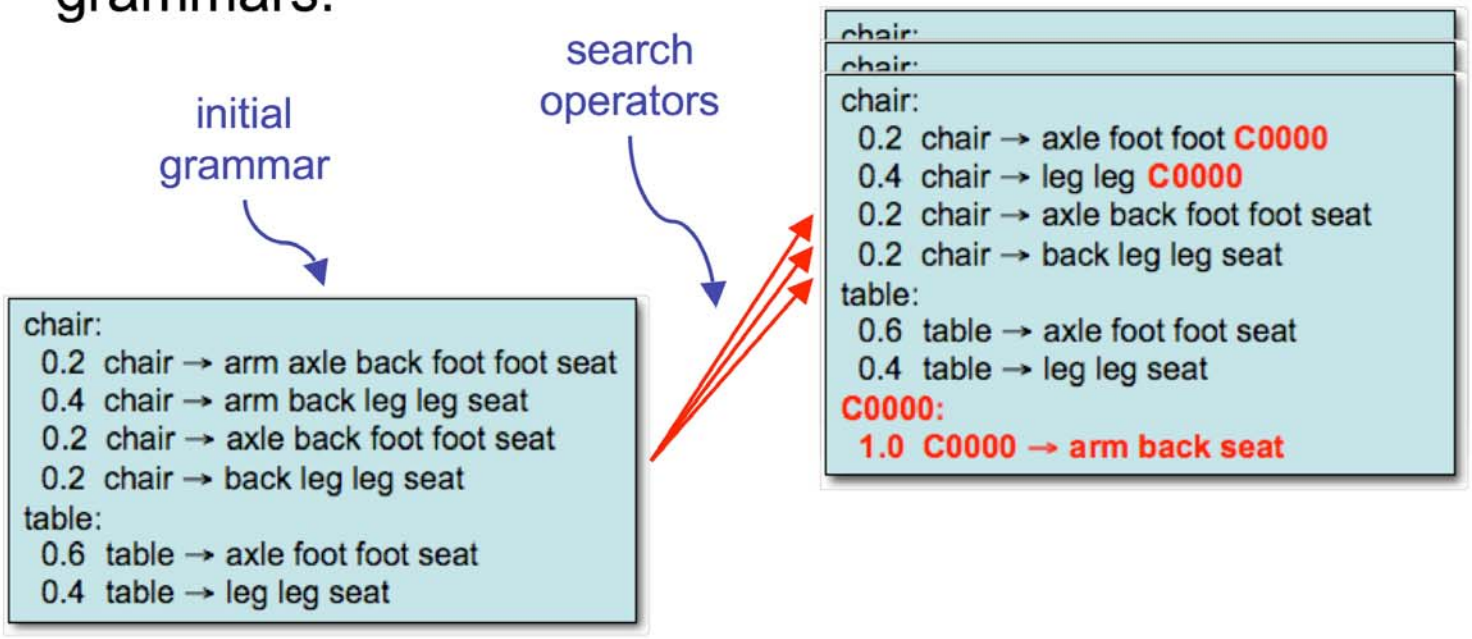
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



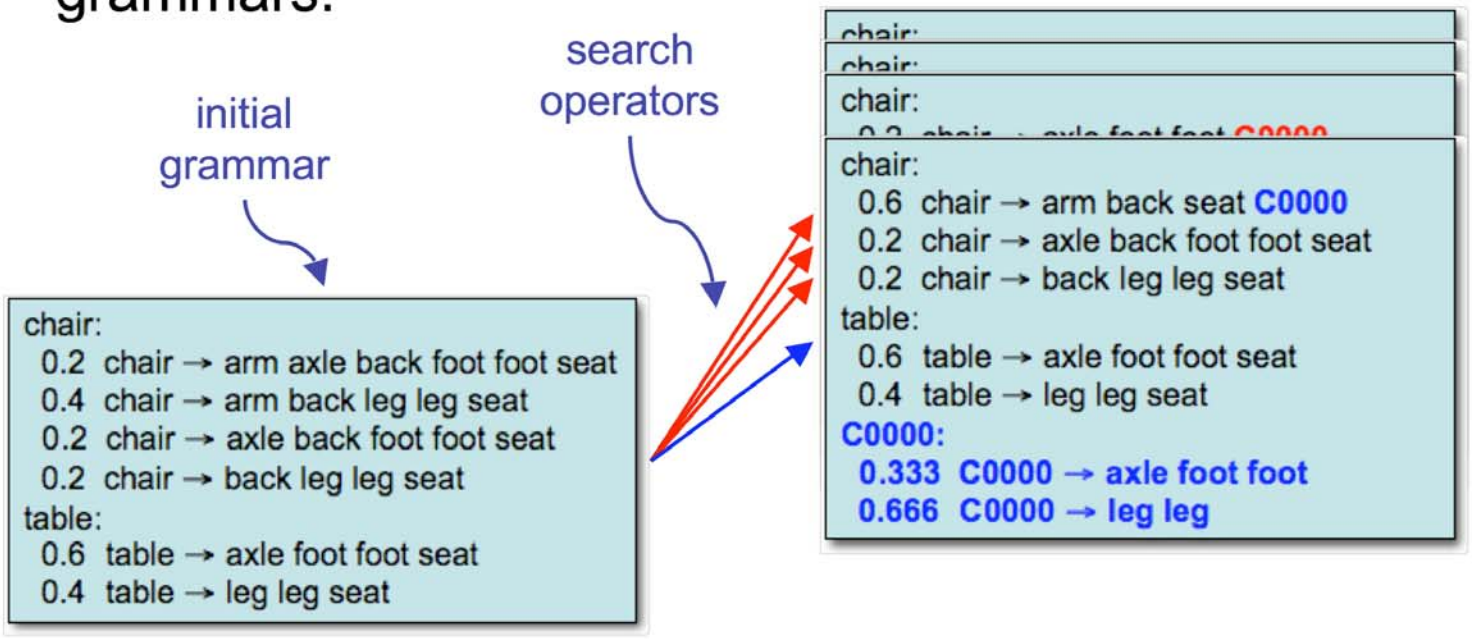
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



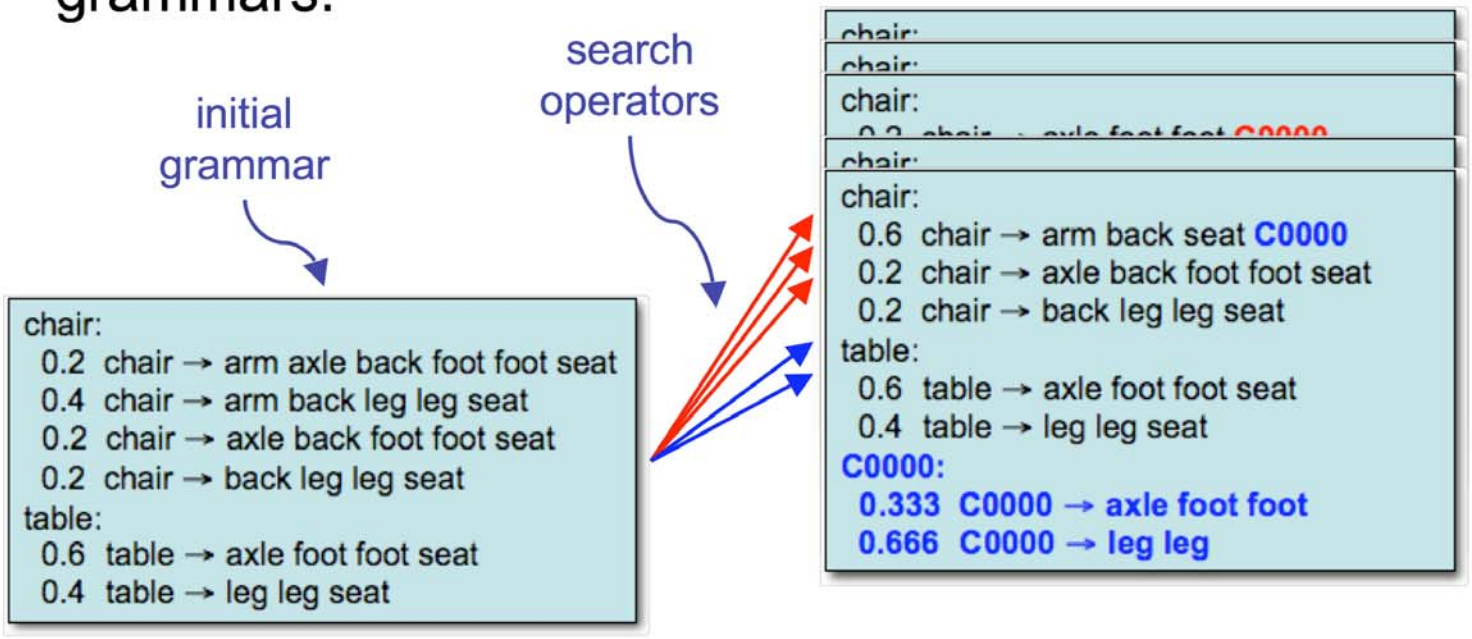
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



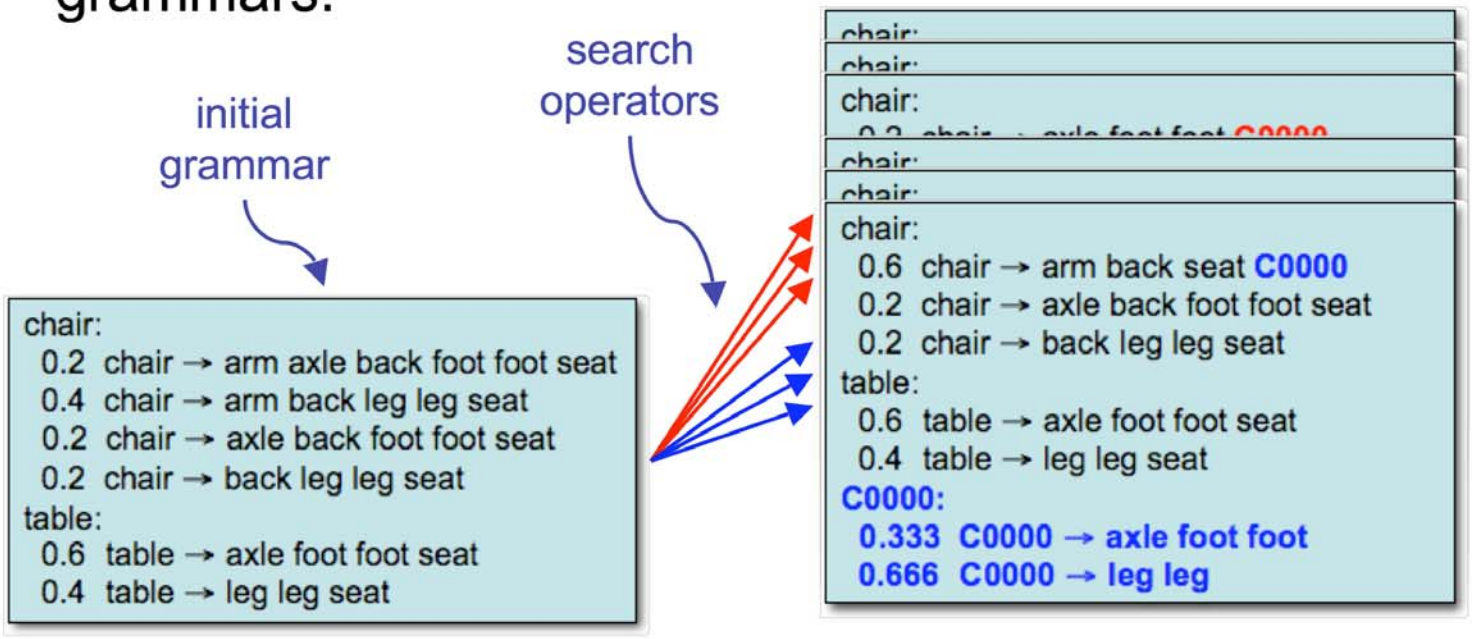
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



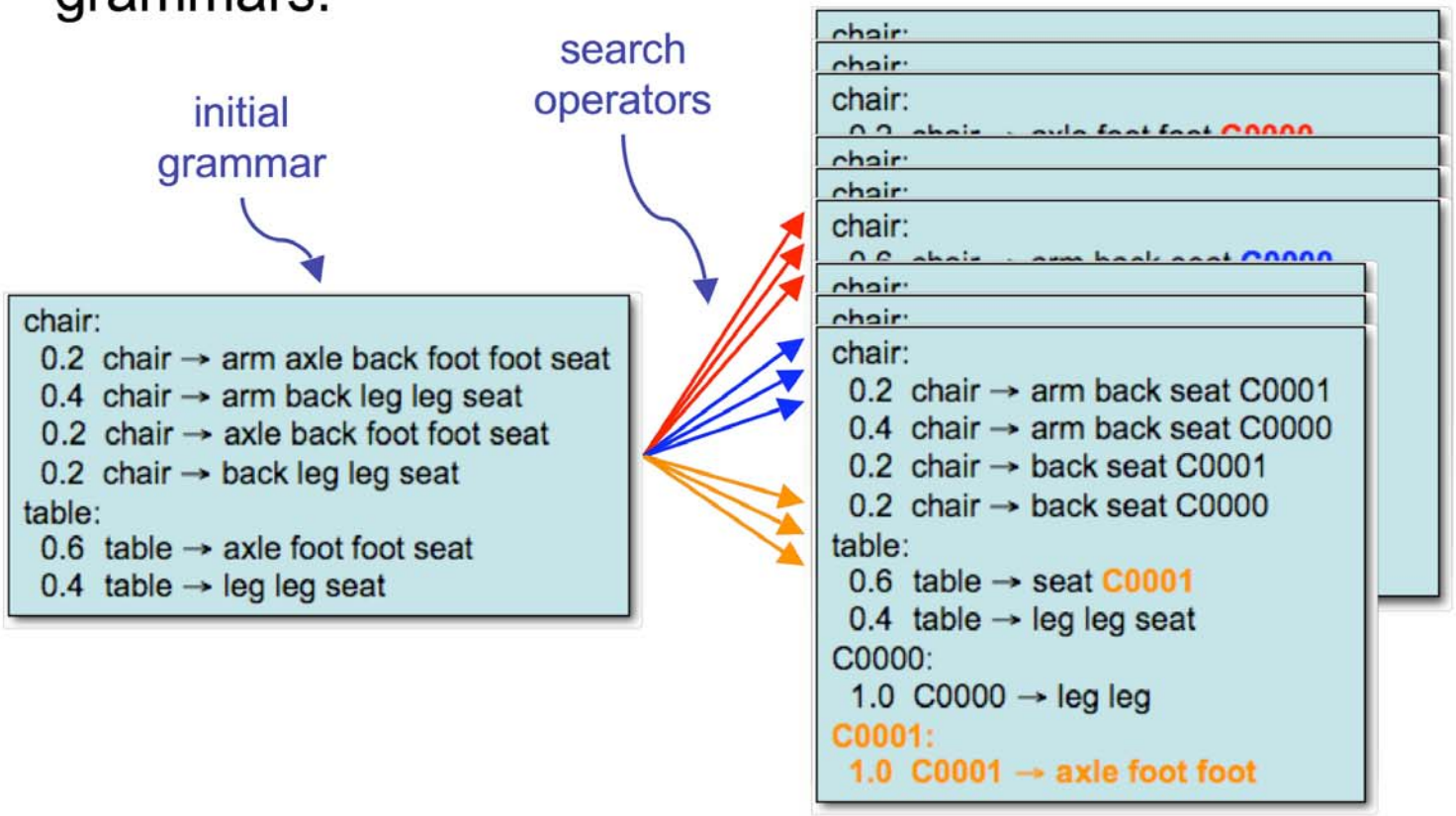
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



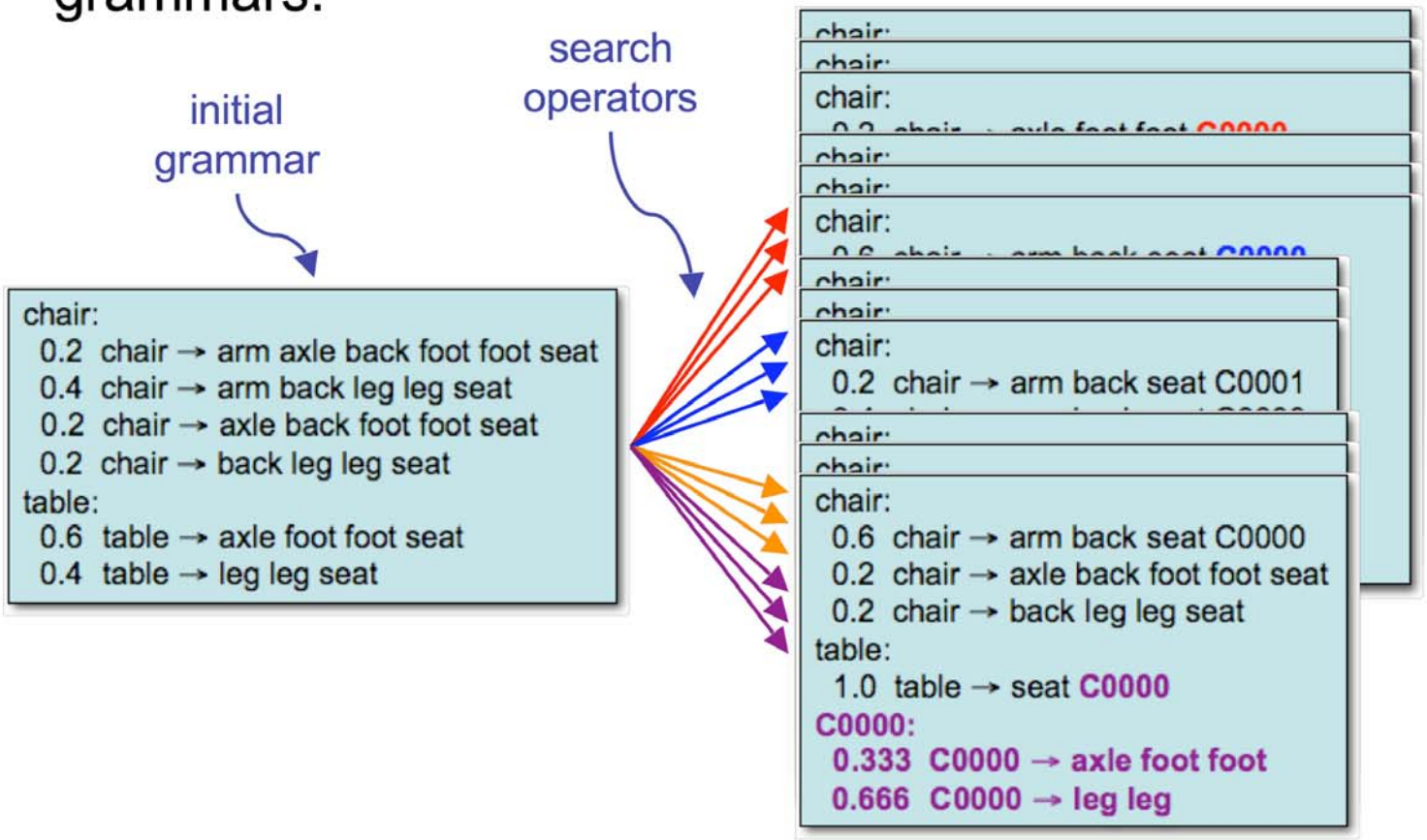
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



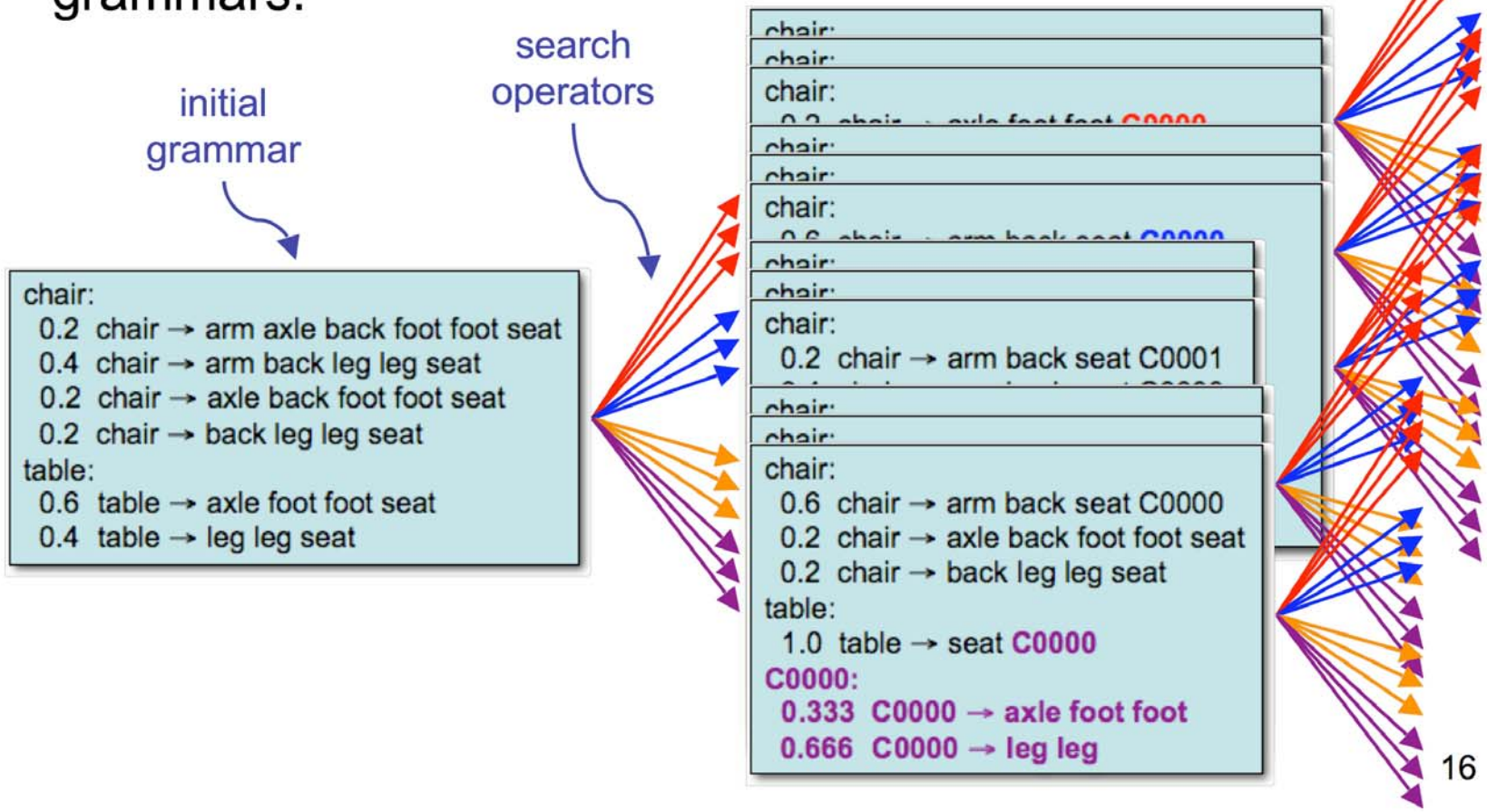
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



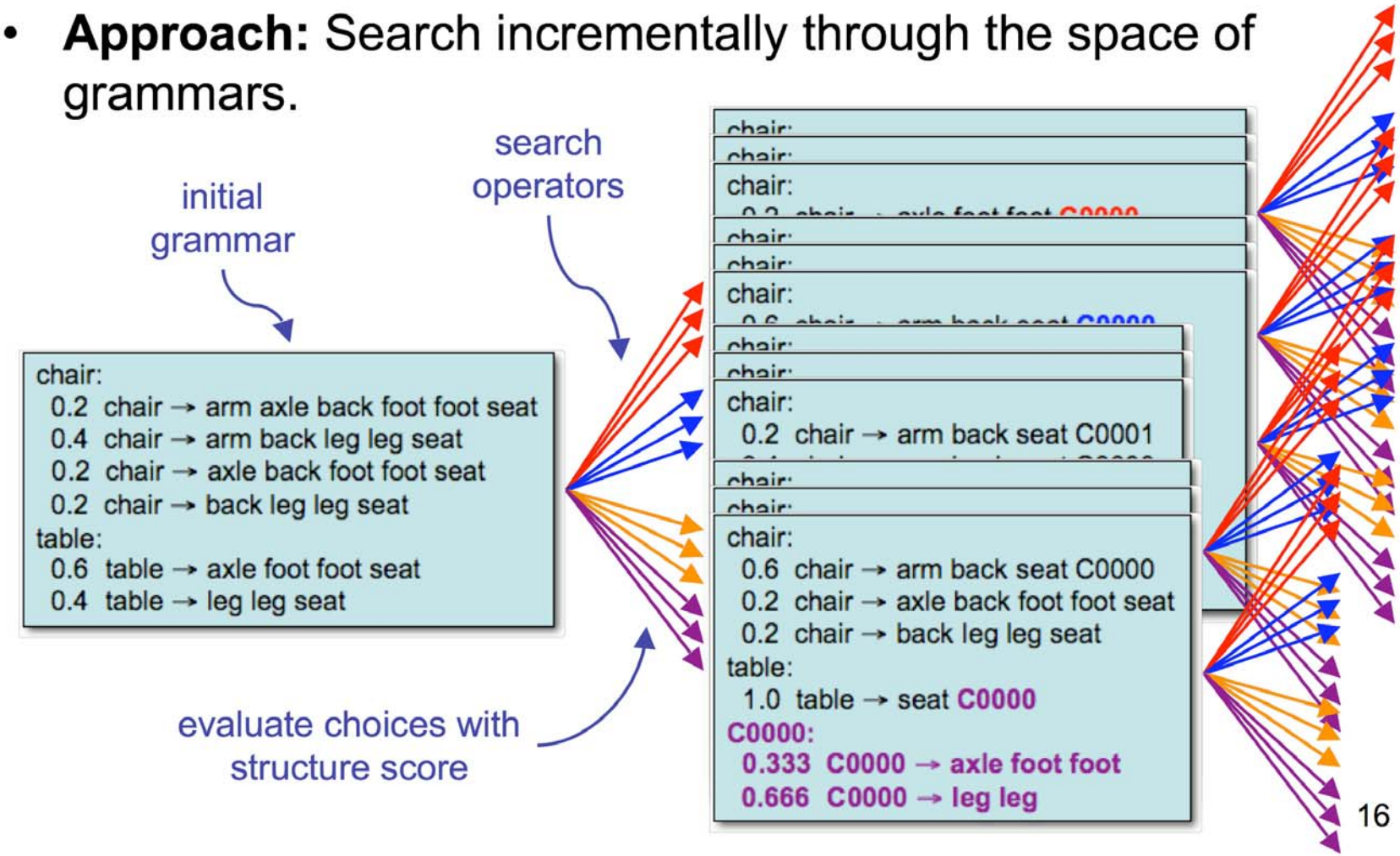
Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



Structure Learning in PGGs

- **Goal:** Find compact grammar that fits training data well.
- **Approach:** Search incrementally through the space of grammars.



Search Initialization

object:
- 0.5 object → chair
- 0.5 object → table

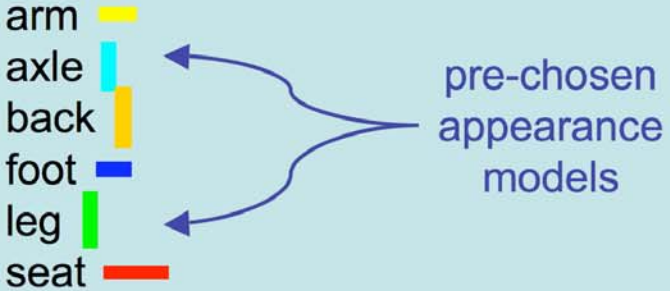
chair:
- 0.2 chair → arm axle back foot foot seat
- 0.4 chair → arm back leg leg seat
- 0.2 chair → axle back foot foot seat
- 0.2 chair → back leg leg seat

table:
- 0.6 table → axle foot foot seat
- 0.4 table → leg leg seat

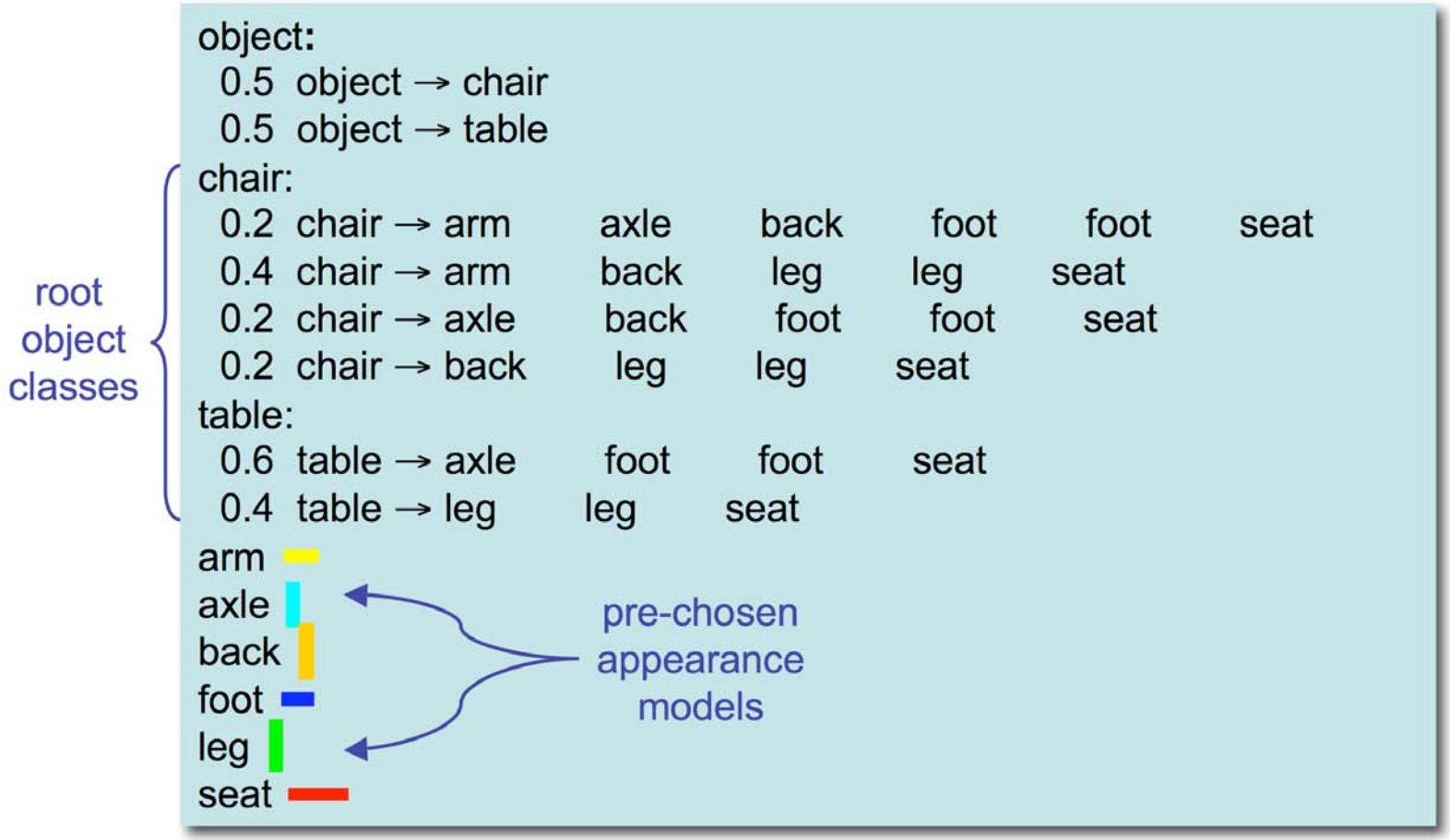
arm
axle
back
foot
leg
seat

Search Initialization

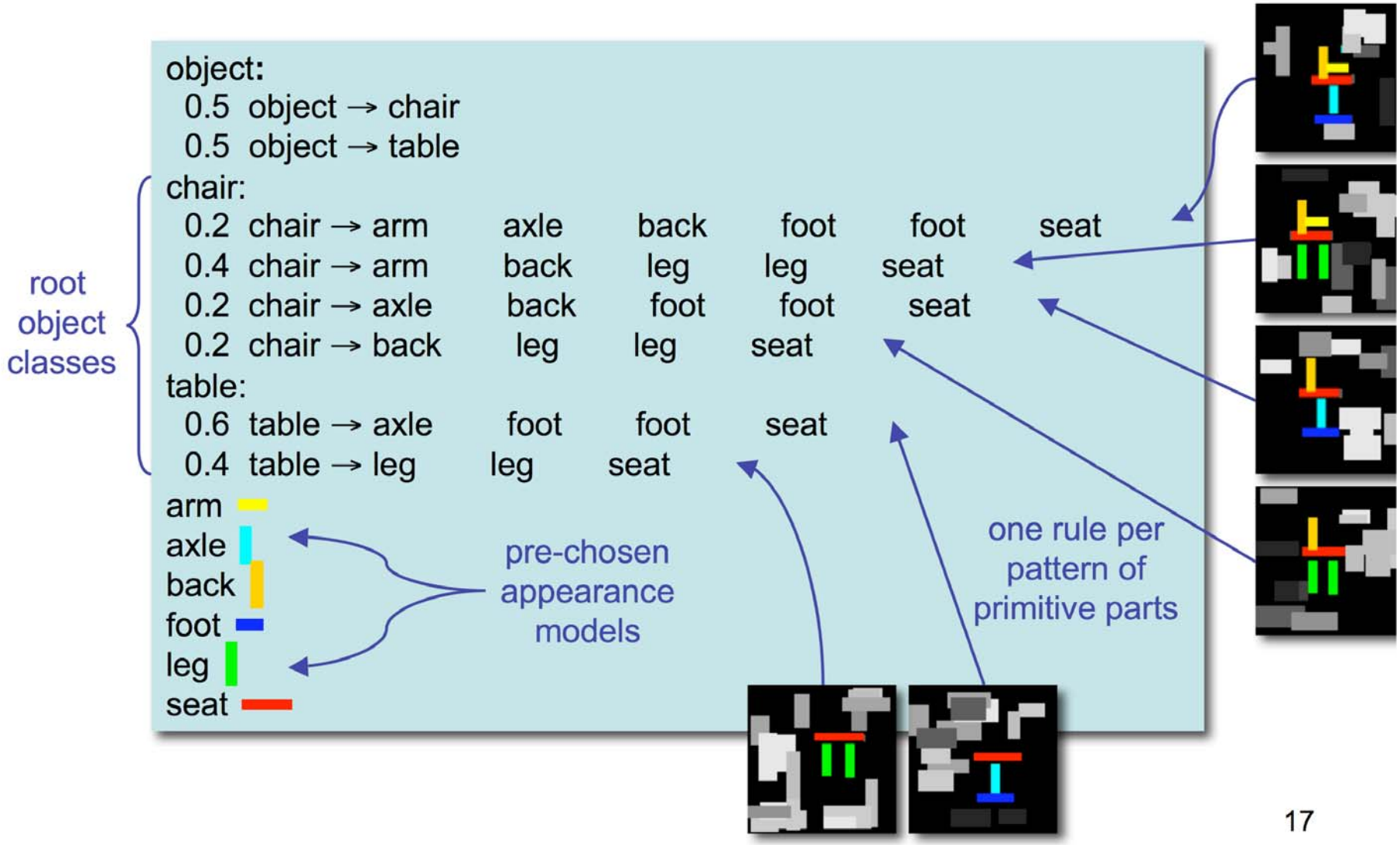
object:
  0.5 object → chair
  0.5 object → table
chair:
  0.2 chair → arm axle back foot foot seat
  0.4 chair → arm back leg leg seat
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat



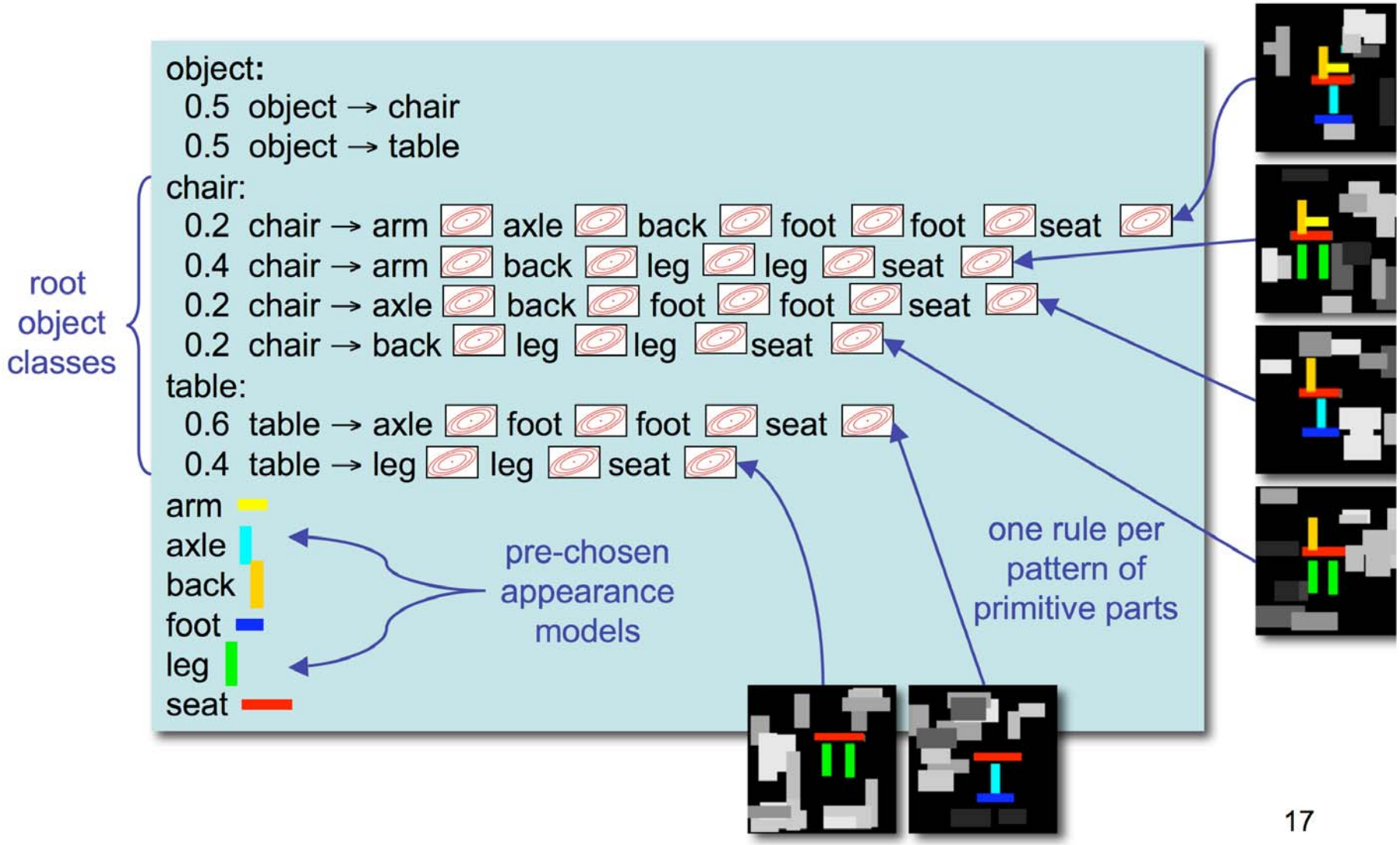
Search Initialization



Search Initialization



Search Initialization



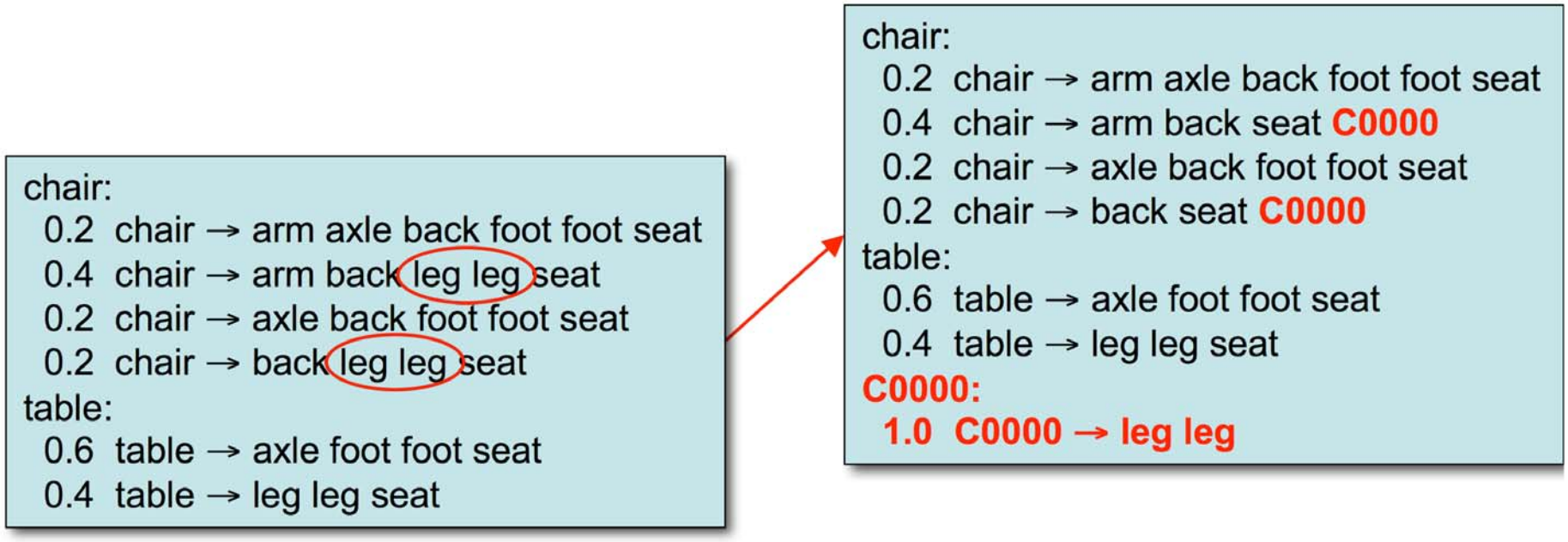
Search Operators: Create AND Part

chair:
  0.2 chair → arm axle back foot foot seat
  0.4 chair → arm back leg leg seat
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat

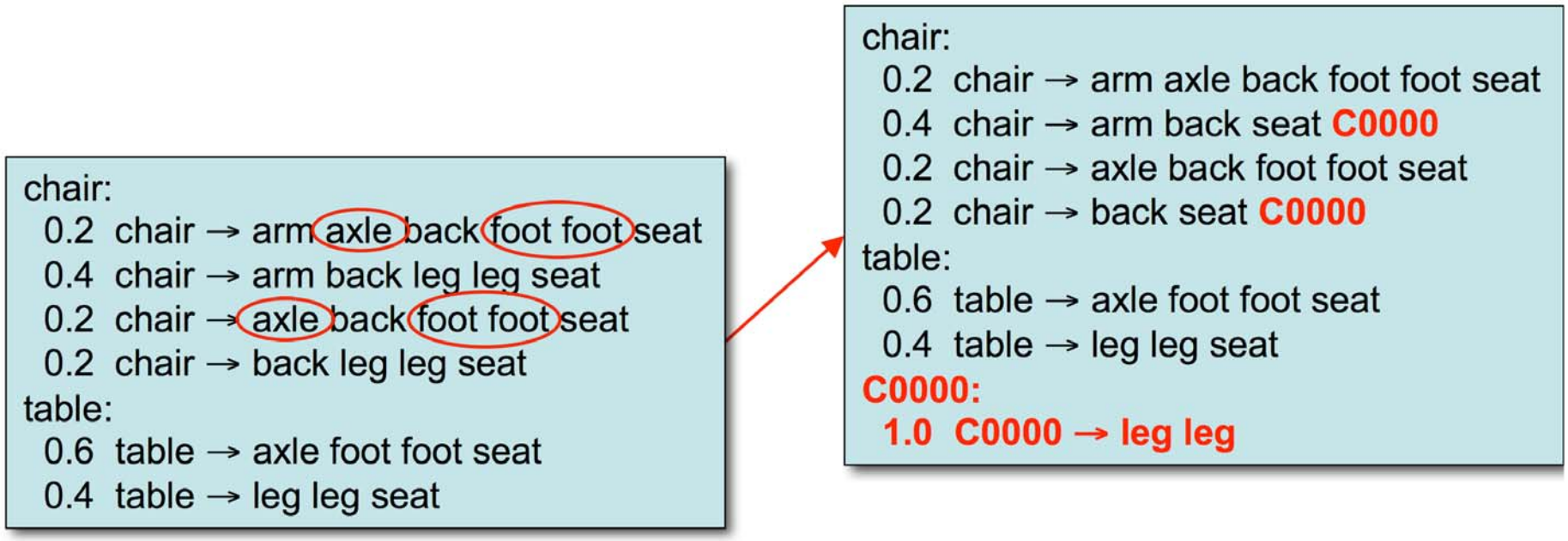
Search Operators: Create AND Part

chair:
  0.2 chair → arm axle back foot foot seat
  0.4 chair → arm back leg leg seat
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat

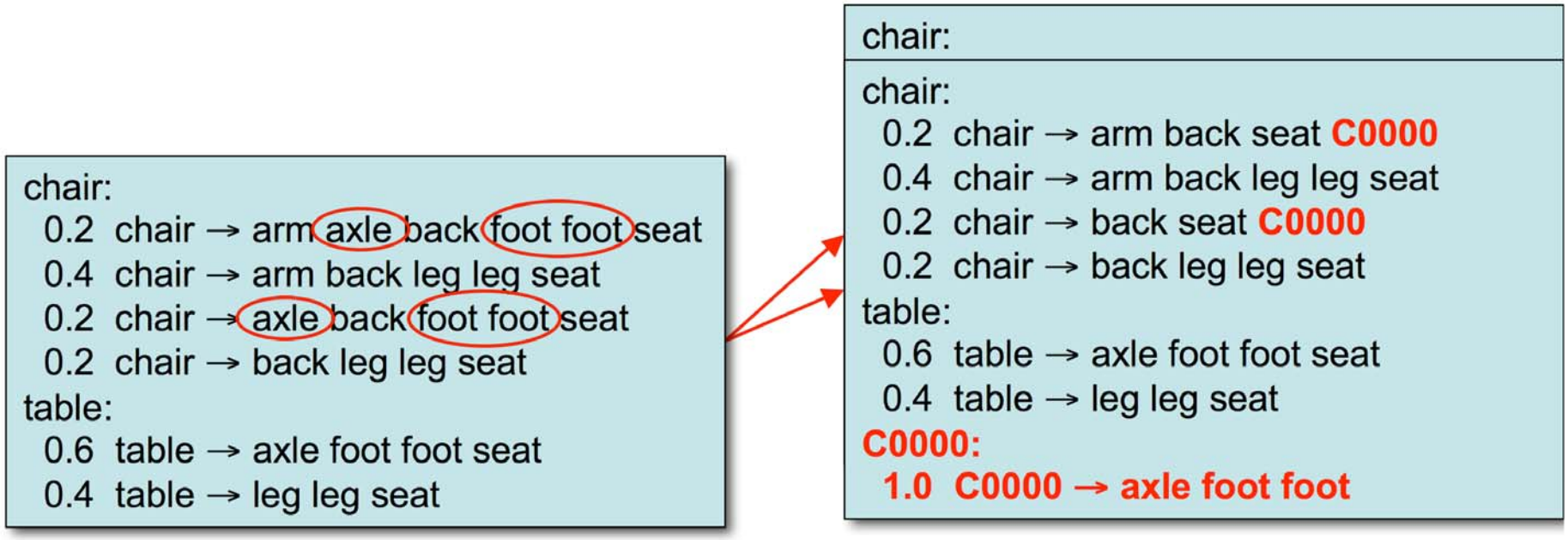
Search Operators: Create AND Part



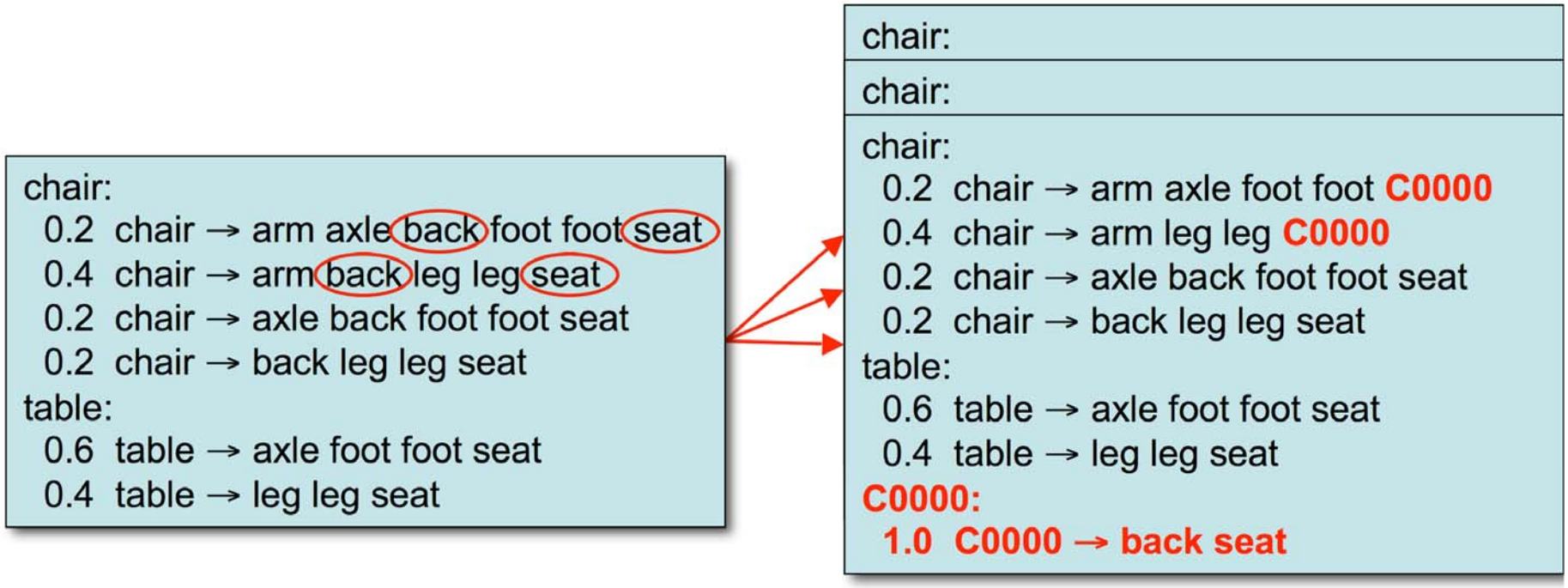
Search Operators: Create AND Part



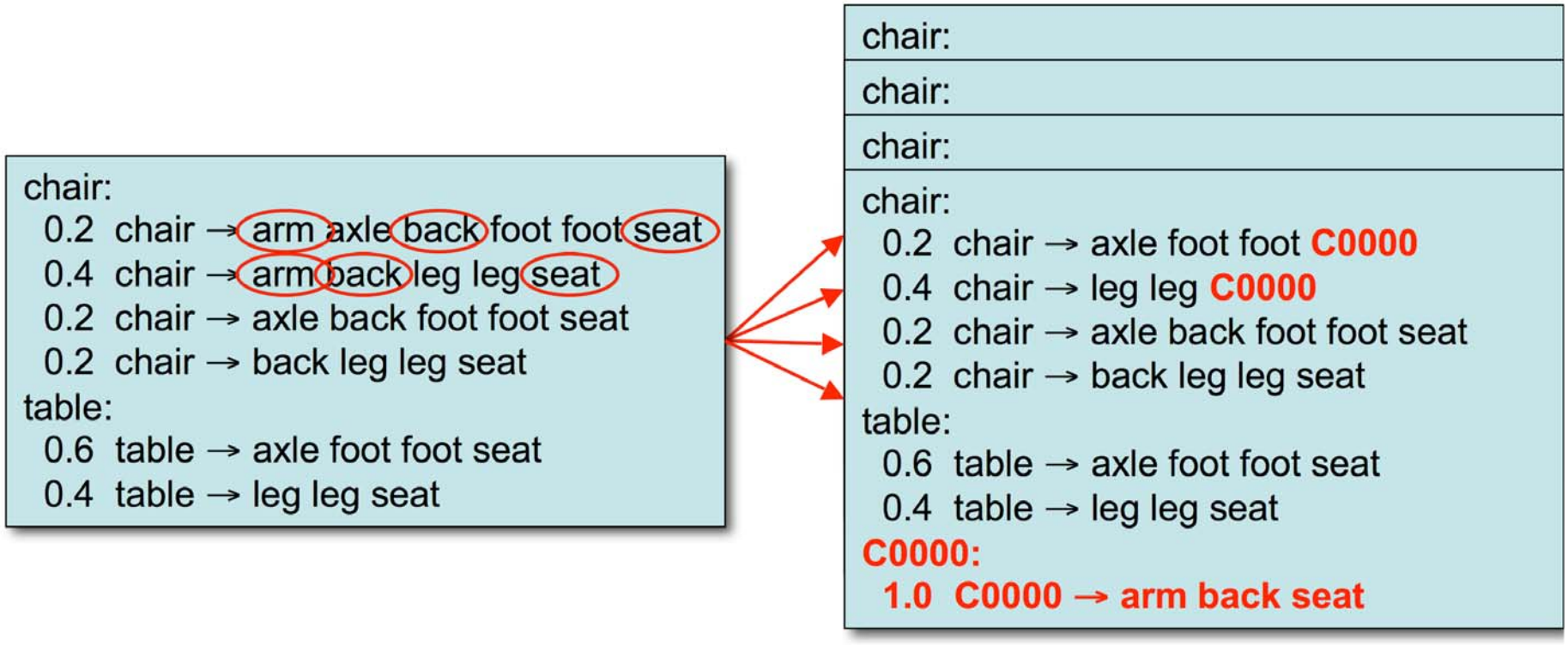
Search Operators: Create AND Part



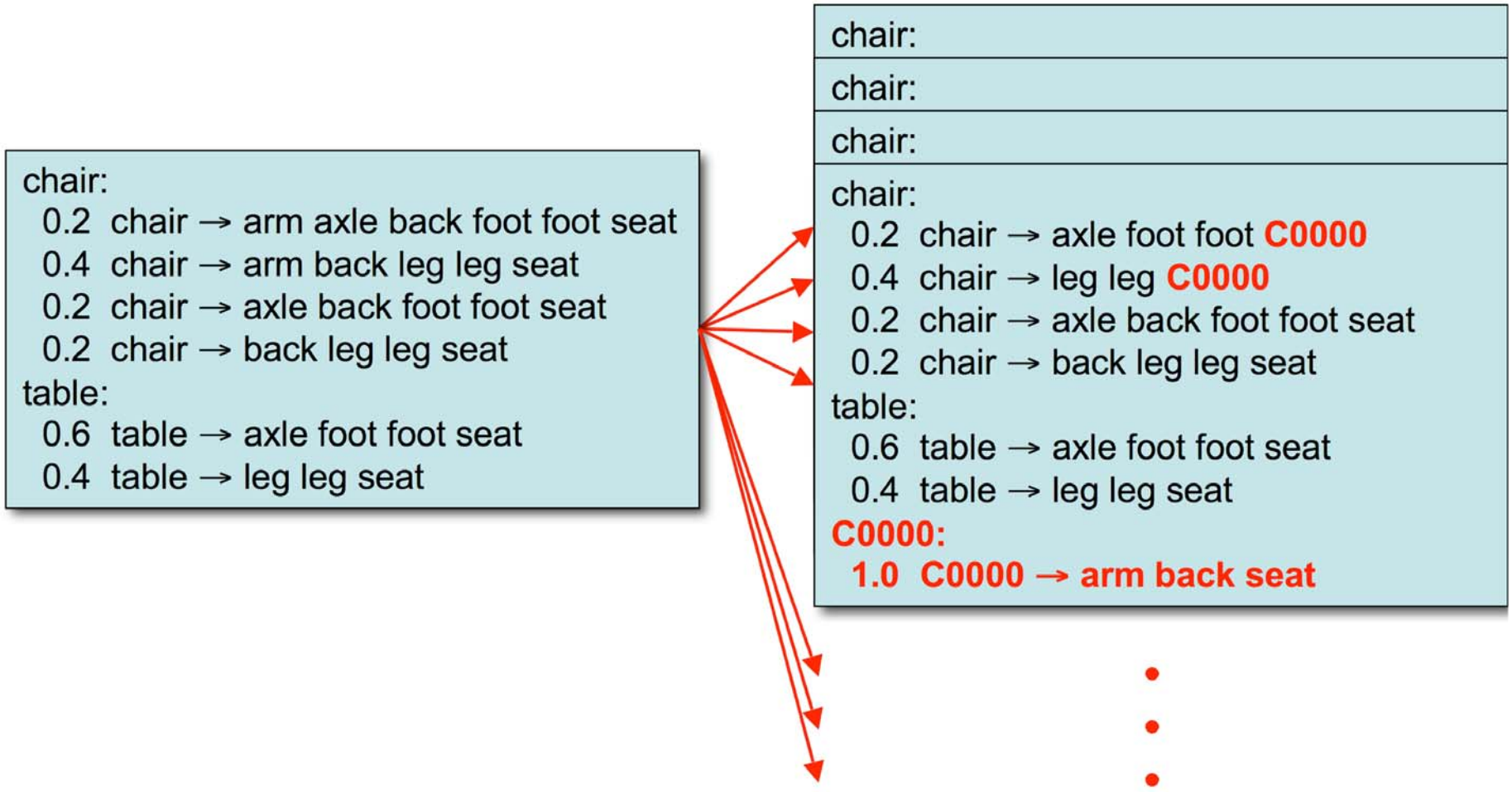
Search Operators: Create AND Part



Search Operators: Create AND Part



Search Operators: Create AND Part



Search Operators: Create OR Part

chair:
- 0.2 chair → arm axle back foot foot seat
- 0.4 chair → arm back leg leg seat
- 0.2 chair → axle back foot foot seat
- 0.2 chair → back leg leg seat

table:
- 0.6 table → axle foot foot seat
- 0.4 table → leg leg seat

Search Operators: Create OR Part

chair:
  0.2 chair → arm axle back foot foot seat
  0.4 chair → arm back leg leg seat
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat

Search Operators: Create OR Part

chair:
- 0.2 chair → arm axle back foot foot seat
- 0.4 chair → arm back leg leg seat
- 0.2 chair → axle back foot foot seat
- 0.2 chair → back leg leg seat

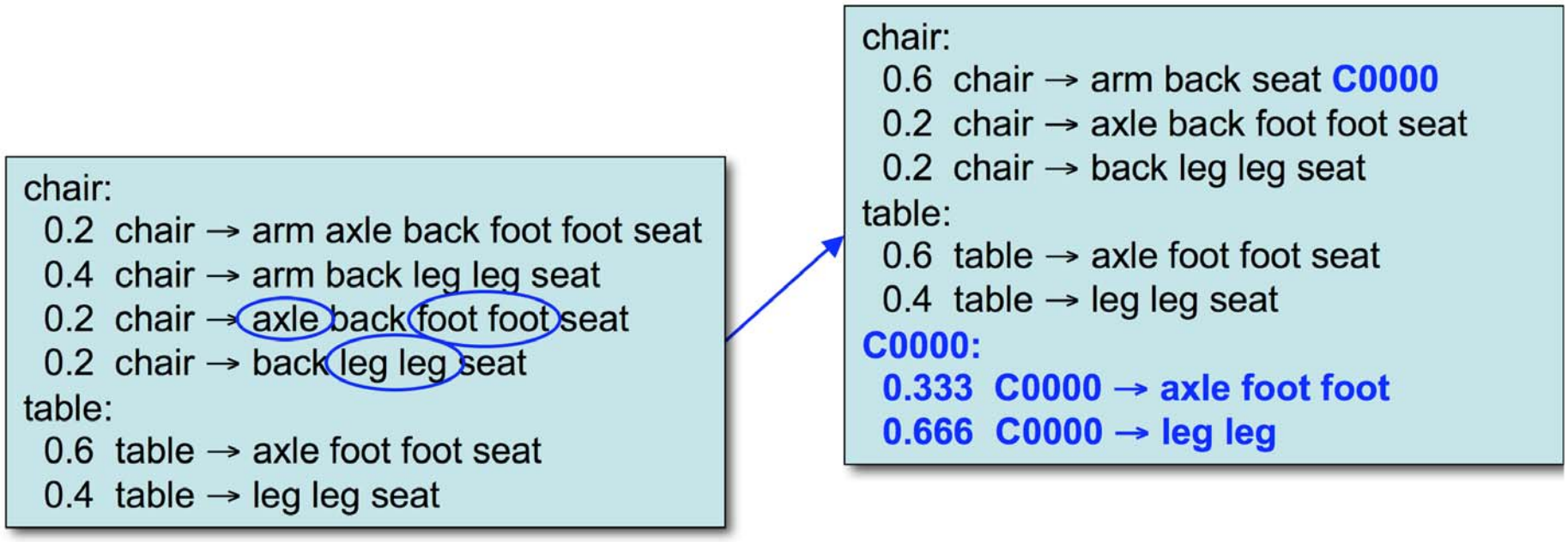
table:
- 0.6 table → axle foot foot seat
- 0.4 table → leg leg seat

chair:
- 0.6 chair → arm back seat **C0000**
- 0.2 chair → axle back foot foot seat
- 0.2 chair → back leg leg seat

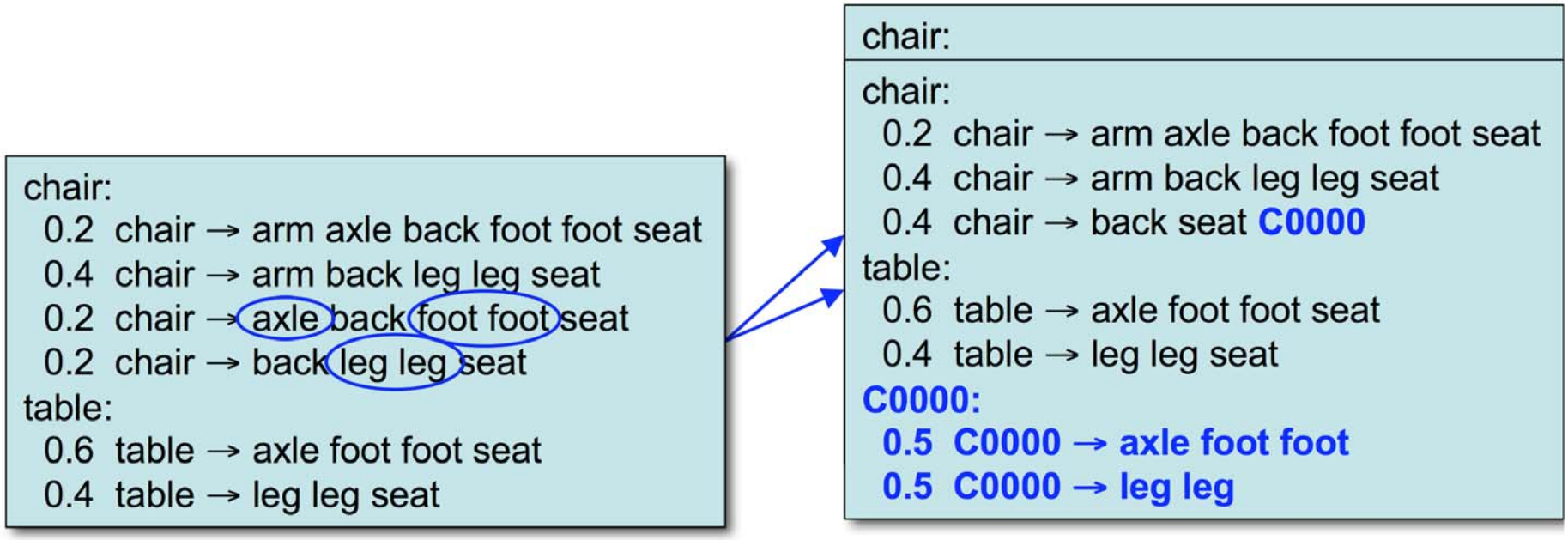
table:
- 0.6 table → axle foot foot seat
- 0.4 table → leg leg seat

C0000:
- **0.333 C0000 → axle foot foot**
- **0.666 C0000 → leg leg**

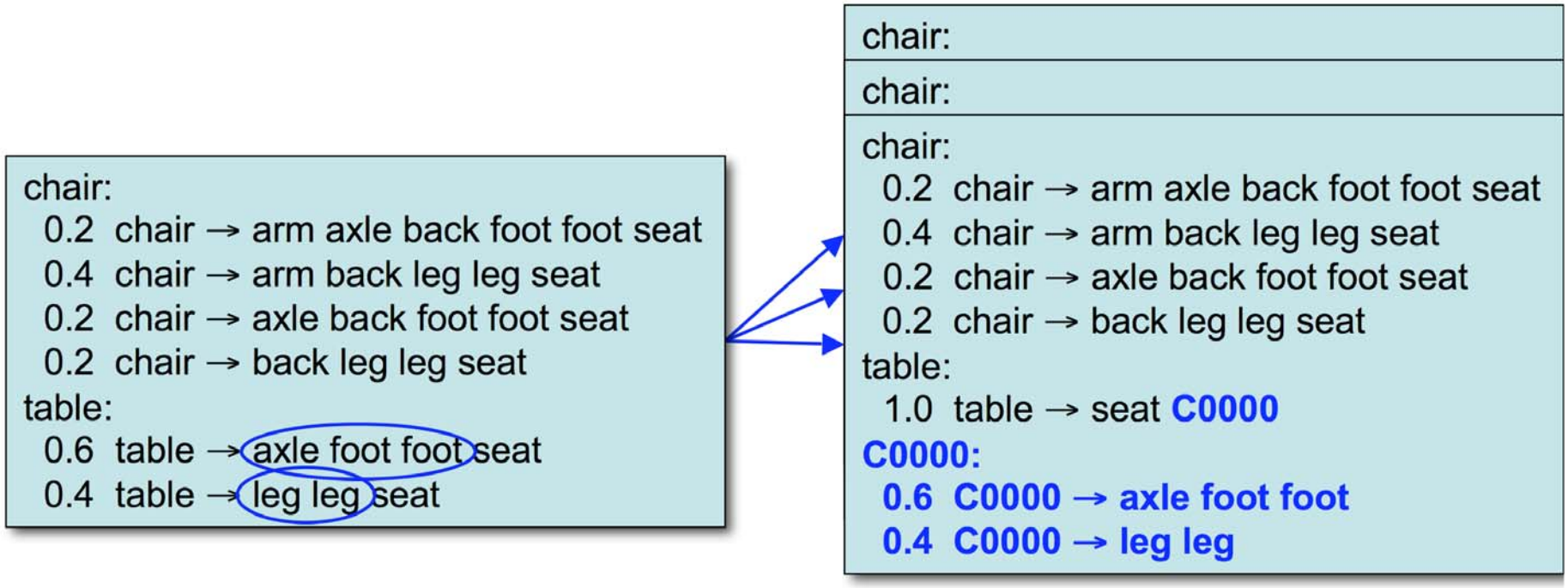
Search Operators: Create OR Part



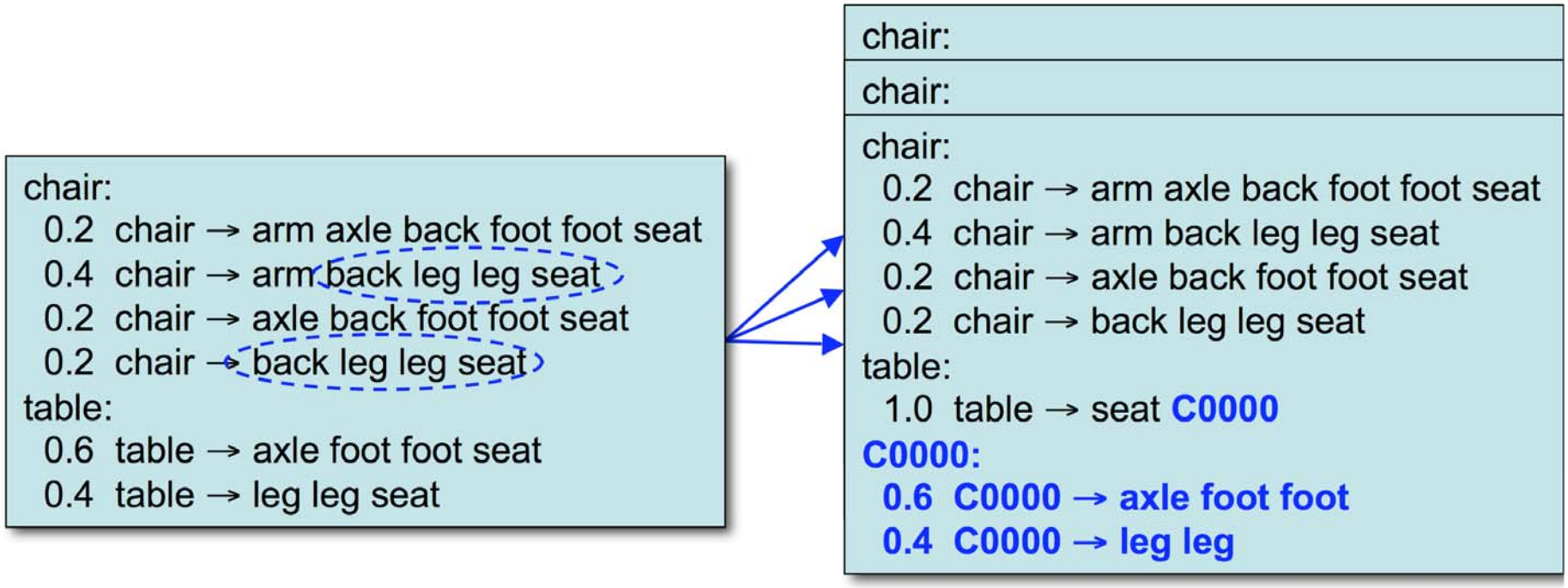
Search Operators: Create OR Part



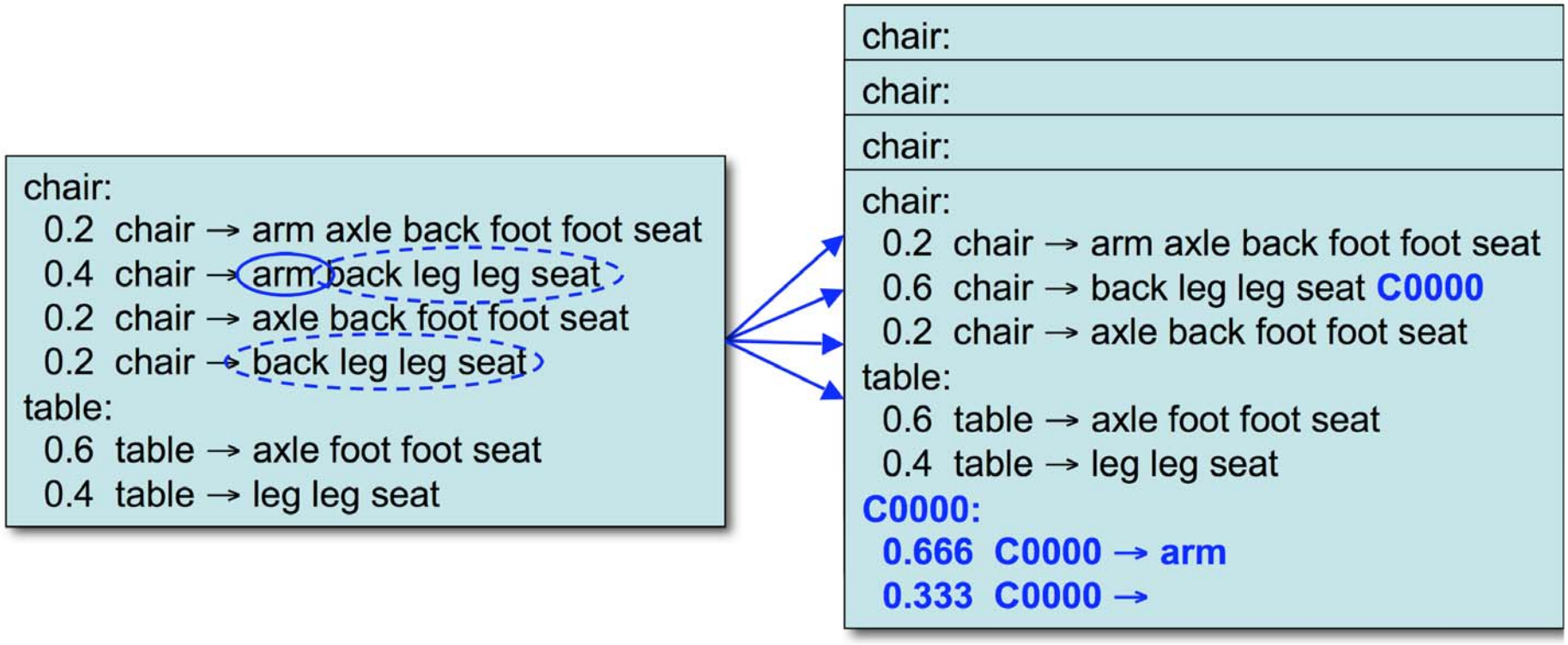
Search Operators: Create OR Part



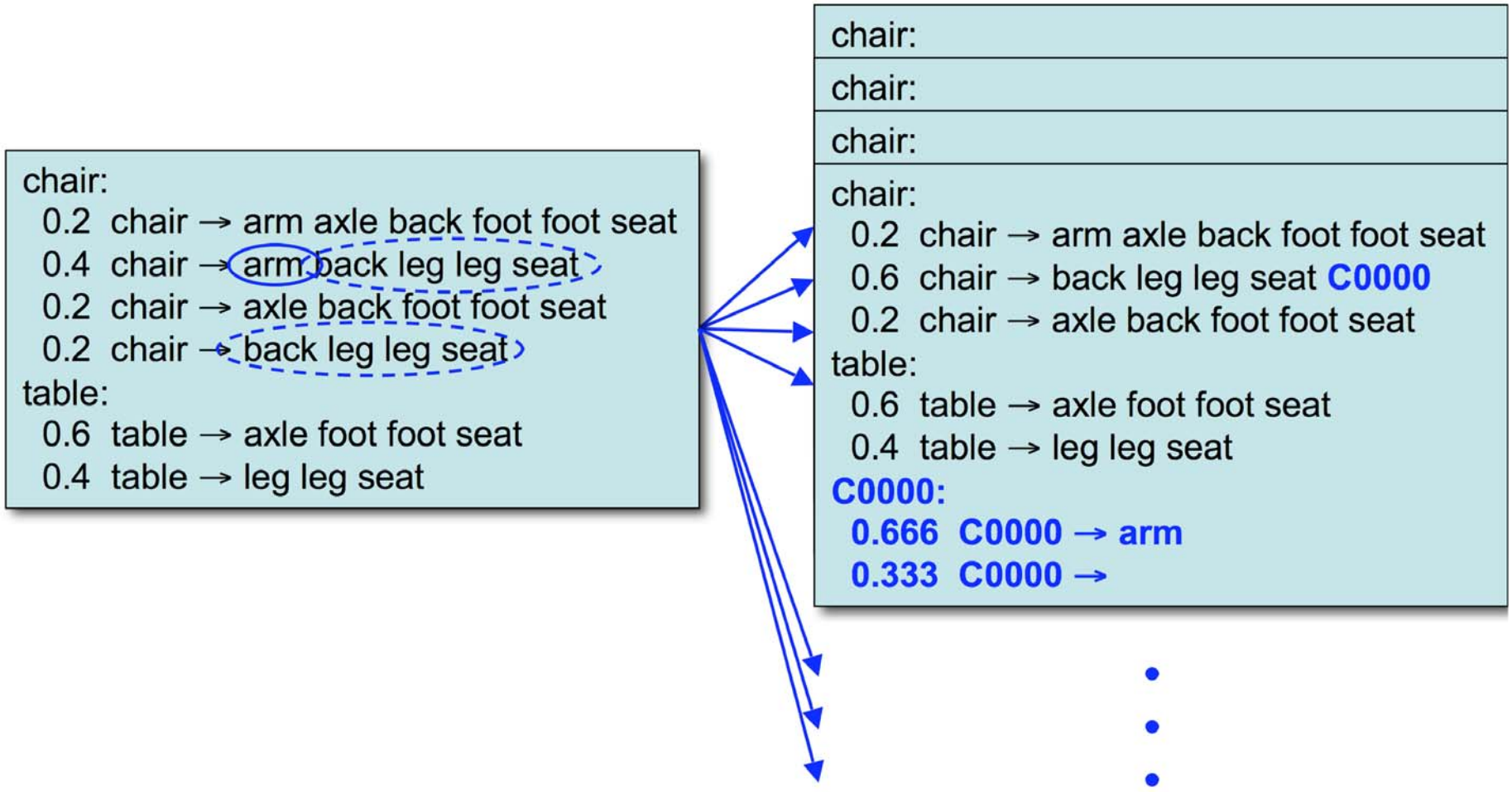
Search Operators: Create OR Part



Search Operators: Create OR Part



Search Operators: Create OR Part



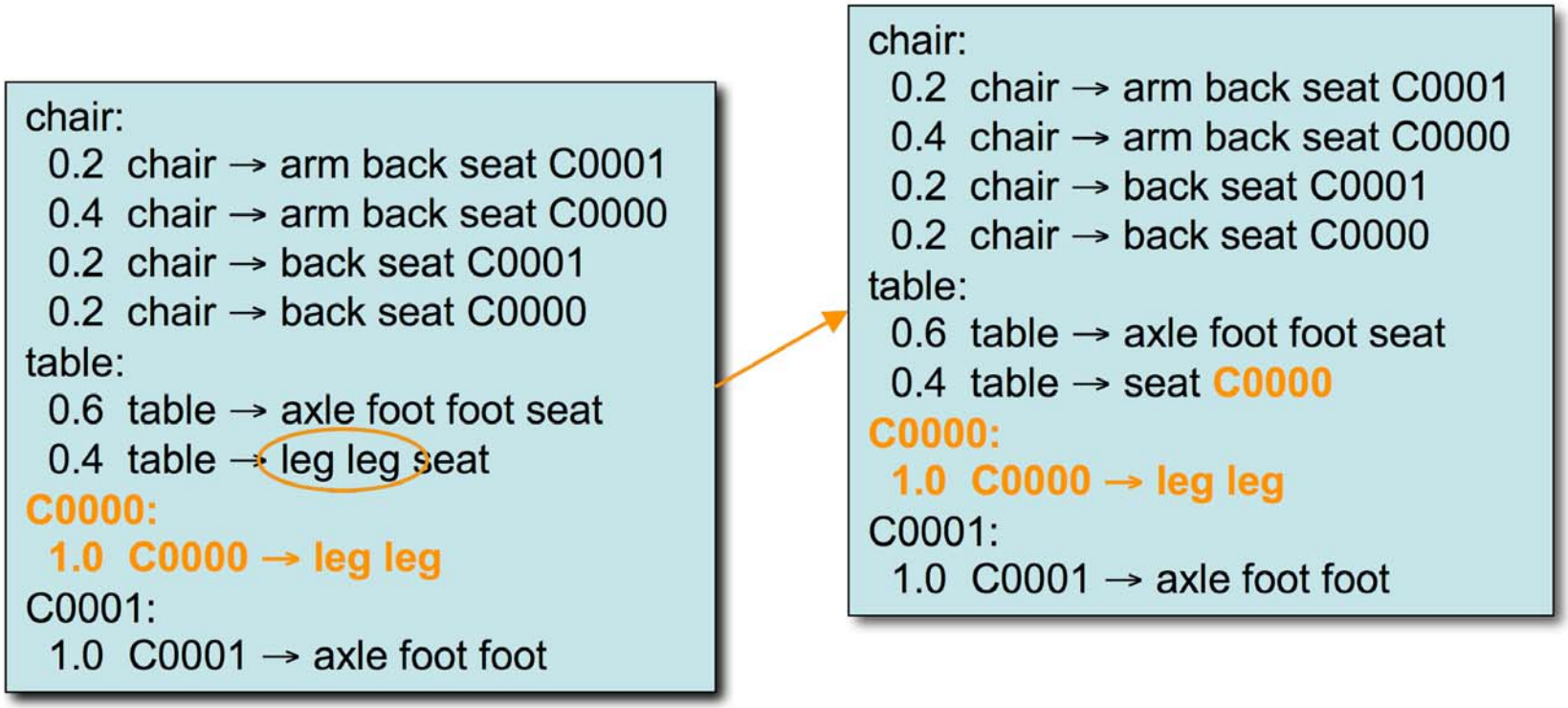
Search Operators: Apply AND Part

chair:
  0.2 chair → arm back seat C0001
  0.4 chair → arm back seat C0000
  0.2 chair → back seat C0001
  0.2 chair → back seat C0000
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat
C0000:
  1.0 C0000 → leg leg
C0001:
  1.0 C0001 → axle foot foot

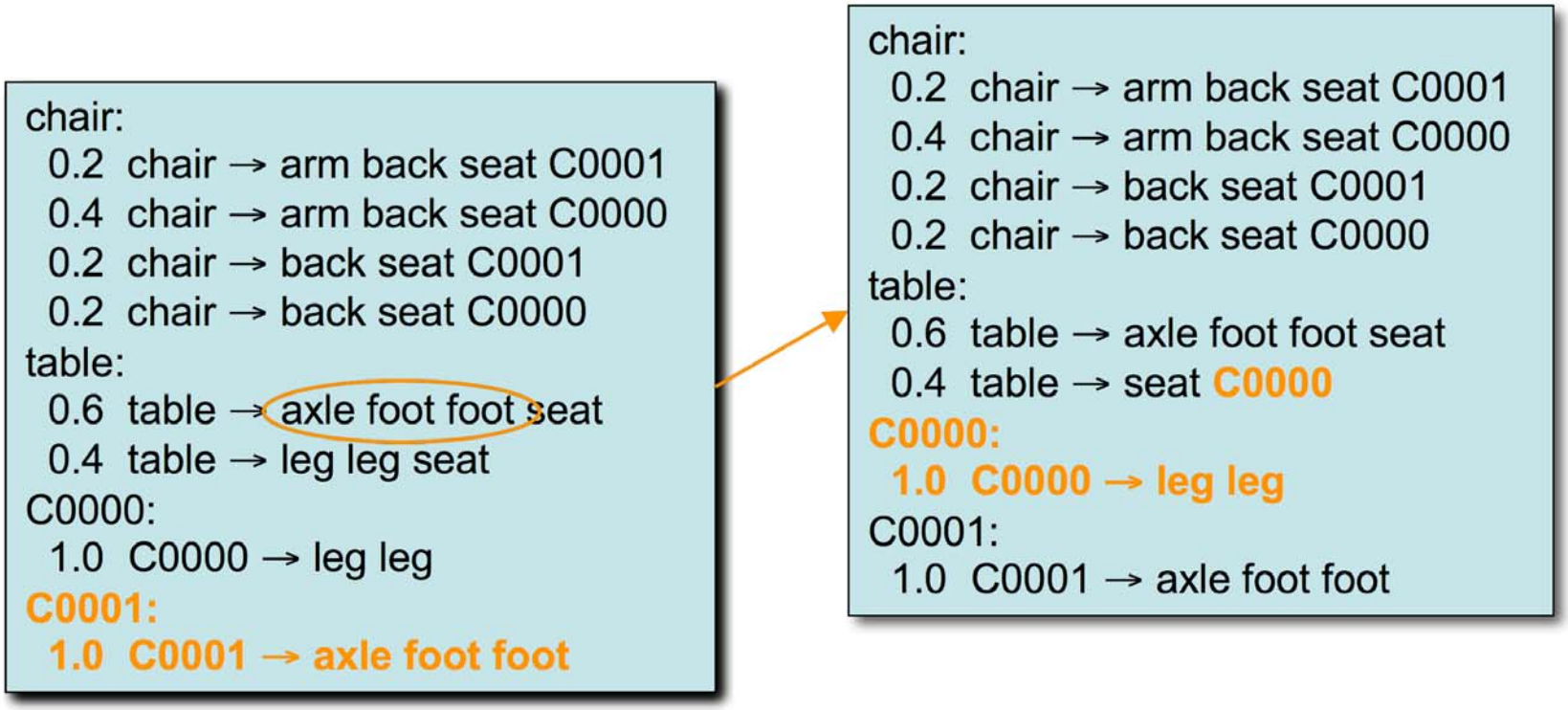
Search Operators: Apply AND Part

chair:
  0.2 chair → arm back seat C0001
  0.4 chair → arm back seat C0000
  0.2 chair → back seat C0001
  0.2 chair → back seat C0000
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat
C0000:
  1.0 C0000 → leg leg
C0001:
  1.0 C0001 → axle foot foot

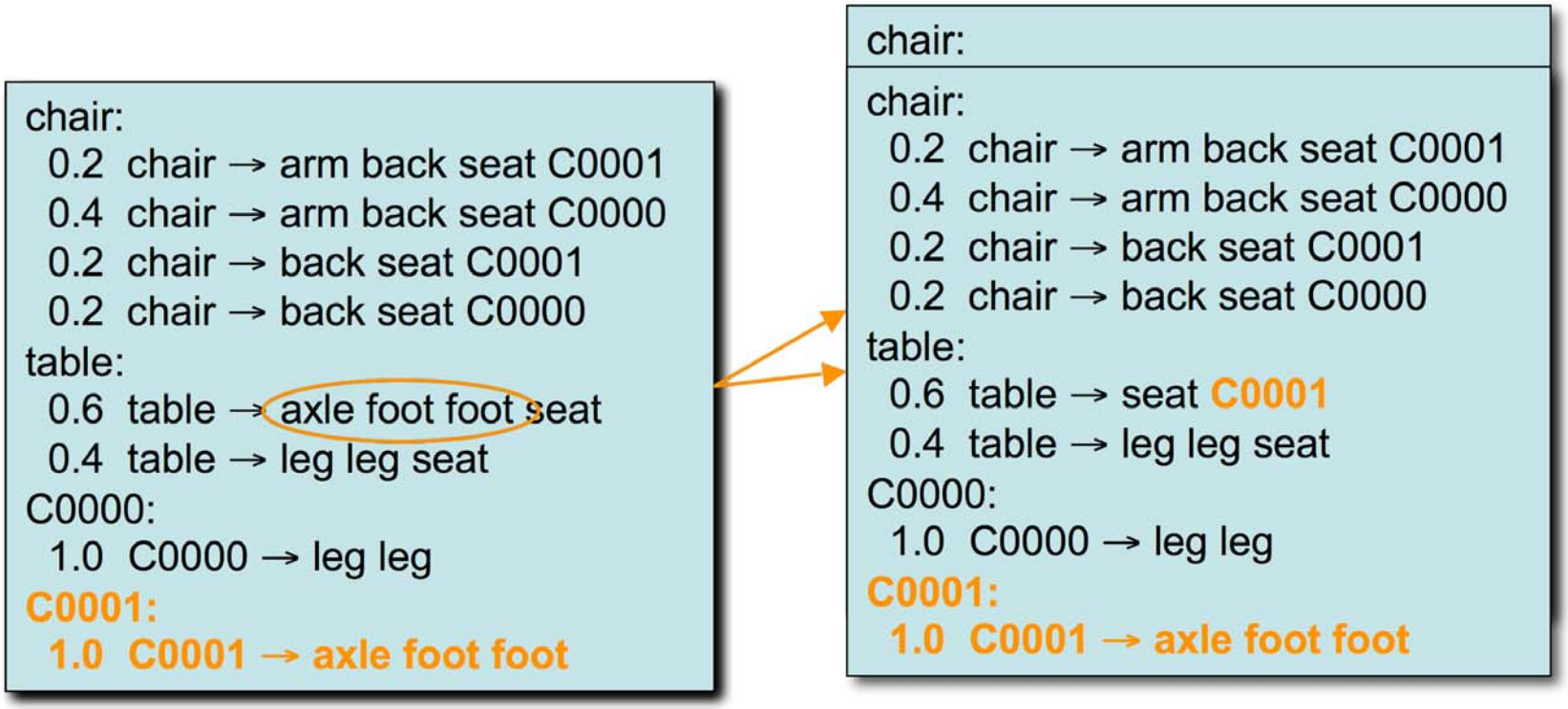
Search Operators: Apply AND Part



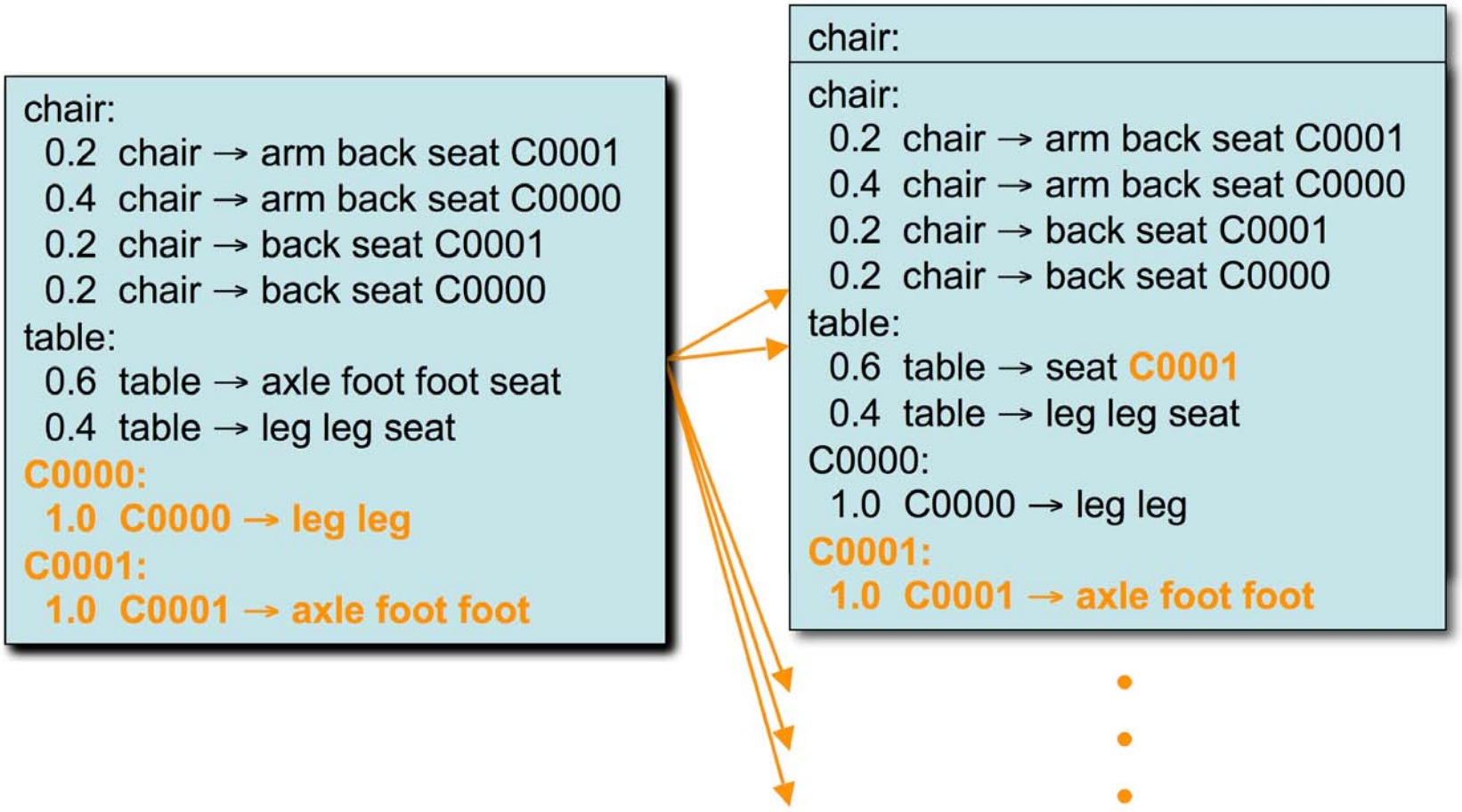
Search Operators: Apply AND Part



Search Operators: Apply AND Part



Search Operators: Apply AND Part



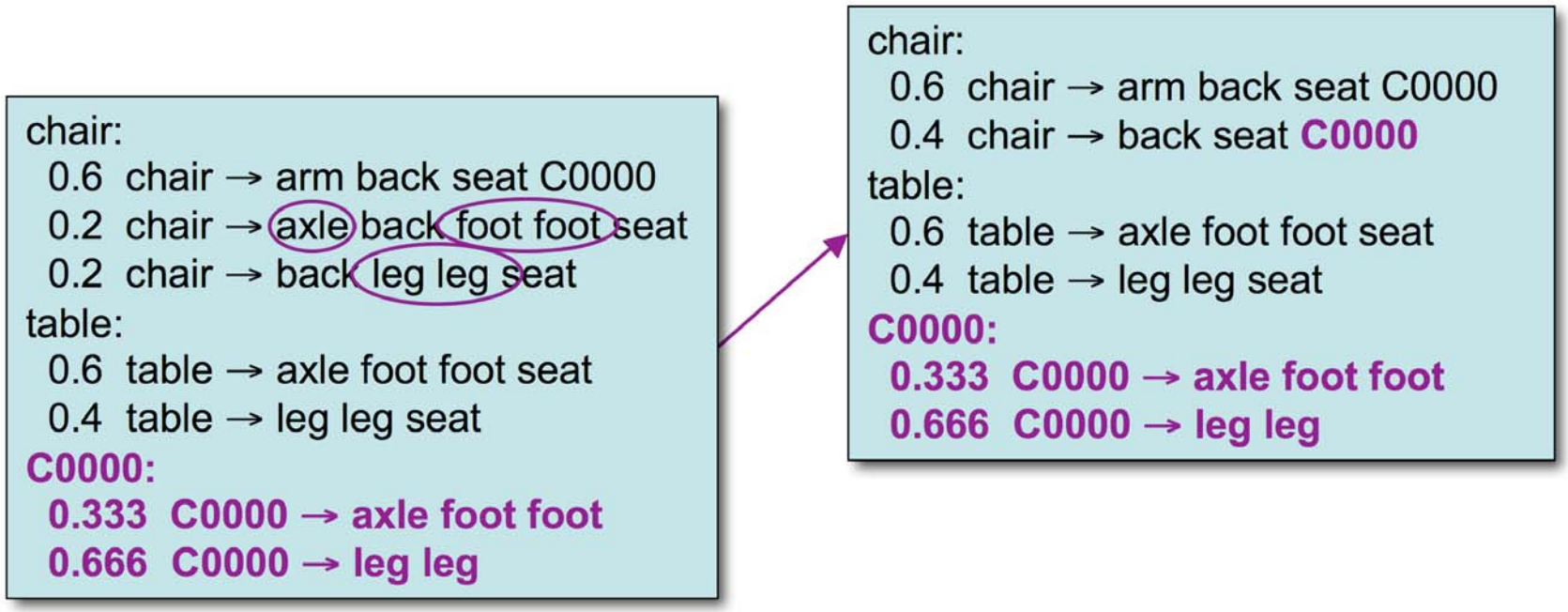
Search Operators: Apply OR Part

chair:
  0.6 chair → arm back seat C0000
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat
C0000:
  0.333 C0000 → axle foot foot
  0.666 C0000 → leg leg

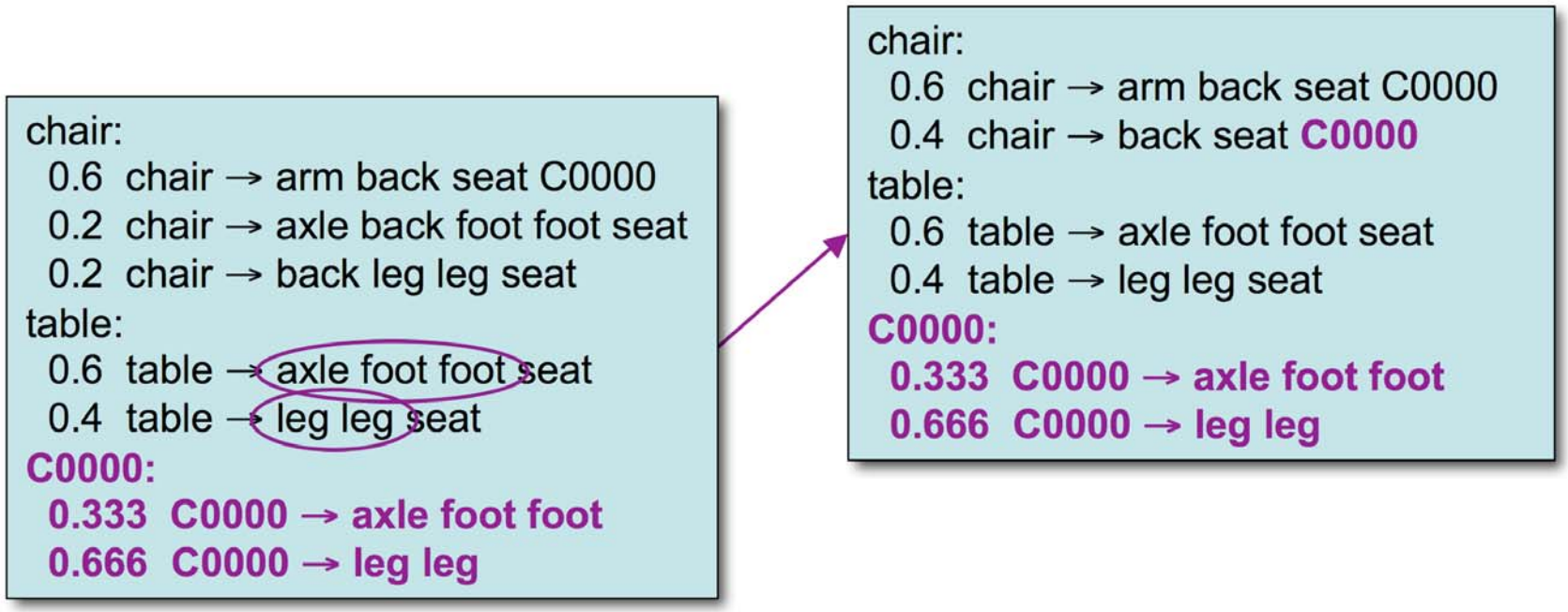
Search Operators: Apply OR Part

chair:
  0.6 chair → arm back seat C0000
  0.2 chair → axle back foot foot seat
  0.2 chair → back leg leg seat
table:
  0.6 table → axle foot foot seat
  0.4 table → leg leg seat
C0000:
  0.333 C0000 → axle foot foot
  0.666 C0000 → leg leg

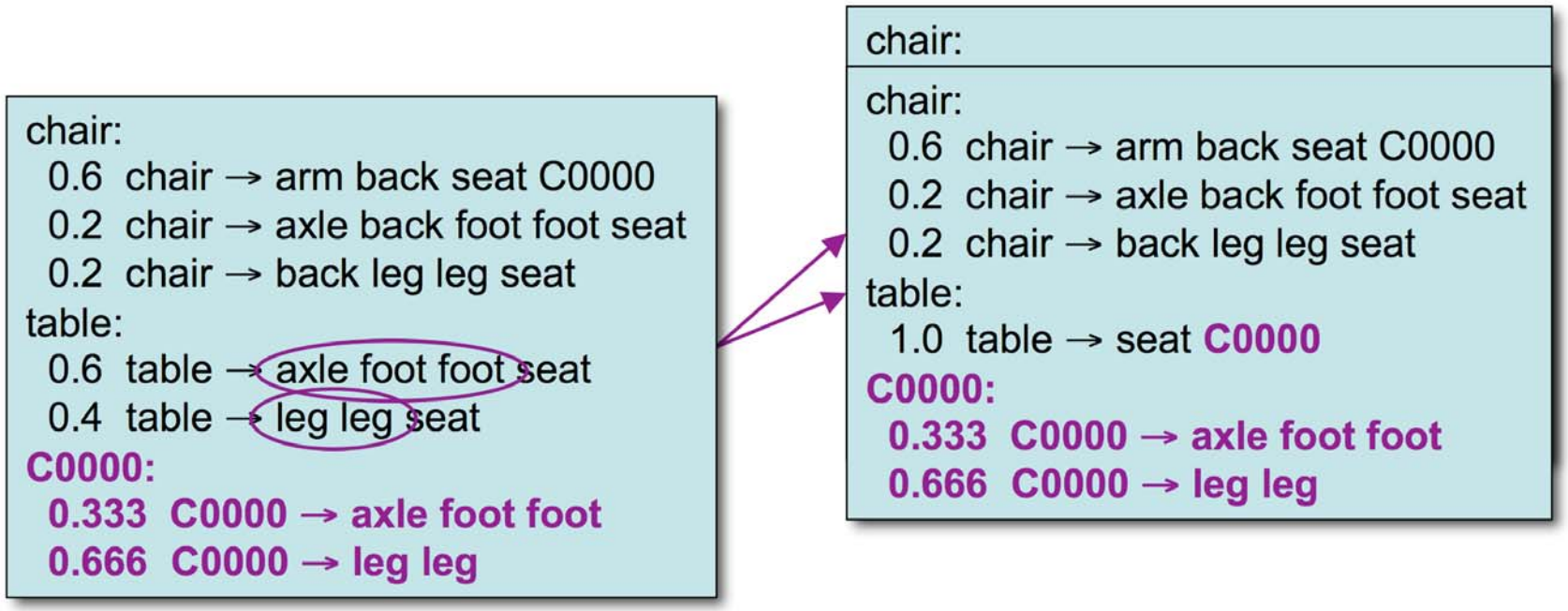
Search Operators: Apply OR Part



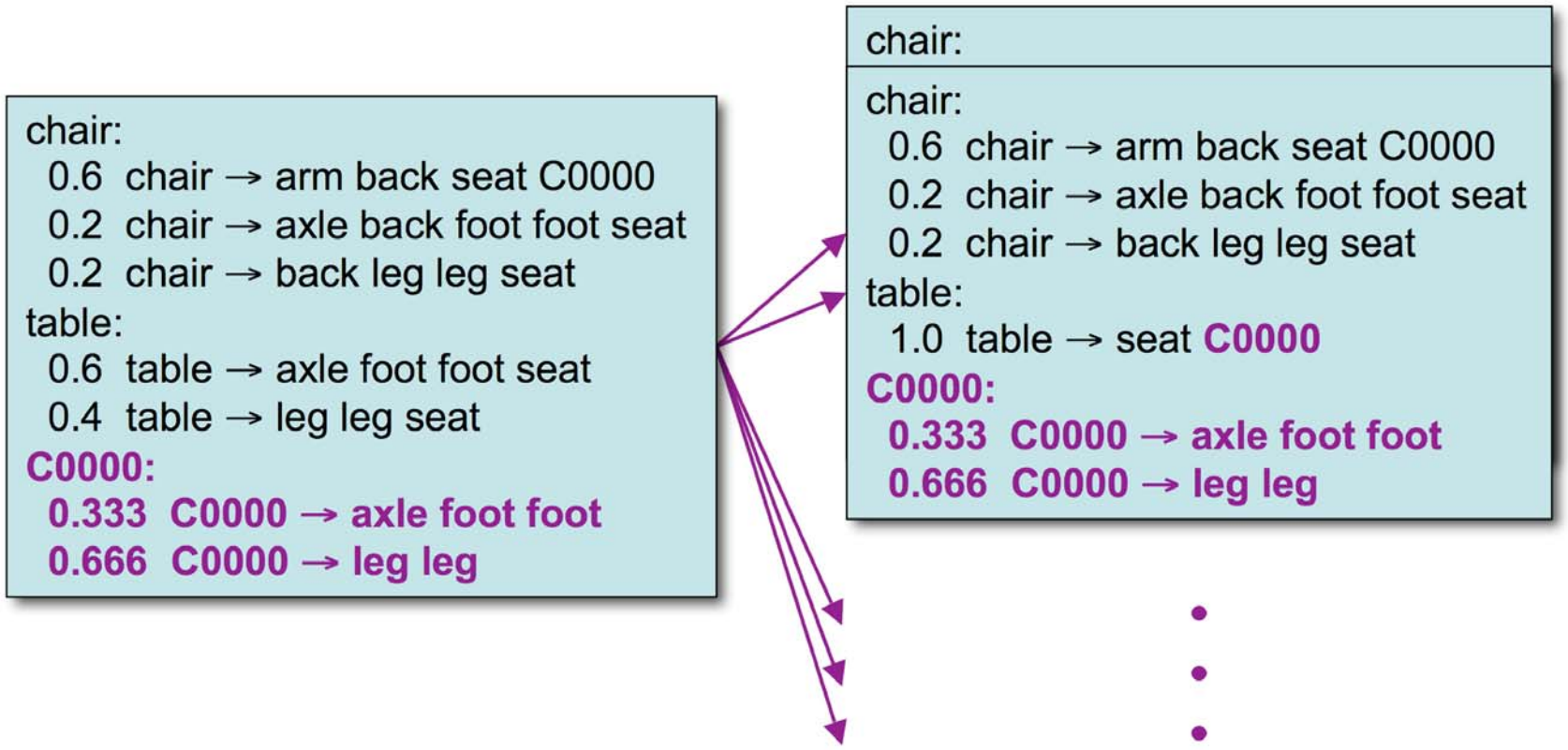
Search Operators: Apply OR Part



Search Operators: Apply OR Part



Search Operators: Apply OR Part



Creating and Evaluating Proposals

Creating and Evaluating Proposals

- Initialize geometry models for new parts to “average” of contributing parts.

Creating and Evaluating Proposals

- Initialize geometry models for new parts to "average" of contributing parts.
- Structure score: $\text{score}(G;\mathbf{D}) = P(\mathbf{D};G) - \alpha \frac{\log|\mathbf{D}|}{2}\dim(G)$

likelihood of data given grammar

weight

complexity penalty

Creating and Evaluating Proposals

- Initialize geometry models for new parts to "average" of contributing parts.
- Structure score: $$\text{score}(G;\mathbf{D}) = \underbrace{P(\mathbf{D};G)}_{\text{likelihood of data given grammar}} - \alpha \underbrace{\frac{\log|\mathbf{D}|}{2}\dim(G)}_{\text{complexity penalty}}$$

  weight
- Choosing a proposal:

Creating and Evaluating Proposals

- Initialize geometry models for new parts to "average" of contributing parts.
- Structure score: $\text{score}(G;\mathbf{D}) = P(\mathbf{D};G) - \alpha \frac{\log|\mathbf{D}|}{2}\dim(G)$

  likelihood of data given grammar

  weight

  complexity penalty
- Choosing a proposal:
  - Rank all new grammars with structure score, before EM.

Creating and Evaluating Proposals

- Initialize geometry models for new parts to "average" of contributing parts.
- Structure score: $\text{score}(G;\mathbf{D}) = P(\mathbf{D};G) - \alpha \frac{\log|\mathbf{D}|}{2}\dim(G)$

  likelihood of data given grammar

  weight

  complexity penalty
- Choosing a proposal:
  - Rank all new grammars with structure score, before EM.
  - Run EM on grammars, in ranked order.

Creating and Evaluating Proposals

- Initialize geometry models for new parts to "average" of contributing parts.
- Structure score: $\text{score}(G;\mathbf{D}) = \underbrace{P(\mathbf{D};G)}_{\text{likelihood of data given grammar}} - \alpha \underbrace{\frac{\log|\mathbf{D}|}{2}\dim(G)}_{\text{complexity penalty}}$

  weight
- Choosing a proposal:
  - Rank all new grammars with structure score, before EM.
  - Run EM on grammars, in ranked order.
  - Accept first proposal whose post-EM score improves on current grammar score.

Wrench Domain

Wrench Domain

- Ground classes:

Wrench Domain

- Ground classes:

- Localization task.

Wrench Domain

- Ground classes:

- Localization task.

Wrench Domain

- Ground classes:

- Localization task.

Wrench Domain

- Ground classes:

- Localization task.

Wrench Domain

- Ground classes:

- Localization task.

Wrench Domain

- Ground classes:

- Localization task.
- Appearance models: hand-chosen edge intensity templates.

function of correlation coefficient with template

$$\frac{P(I_w \mid cw)}{P(I_w)} = \frac{\overbrace{CC(I_w, T_c)}}{\underbrace{\sum_{I_w'} P(I_w')\, CC(I_w', T_c)}}$$

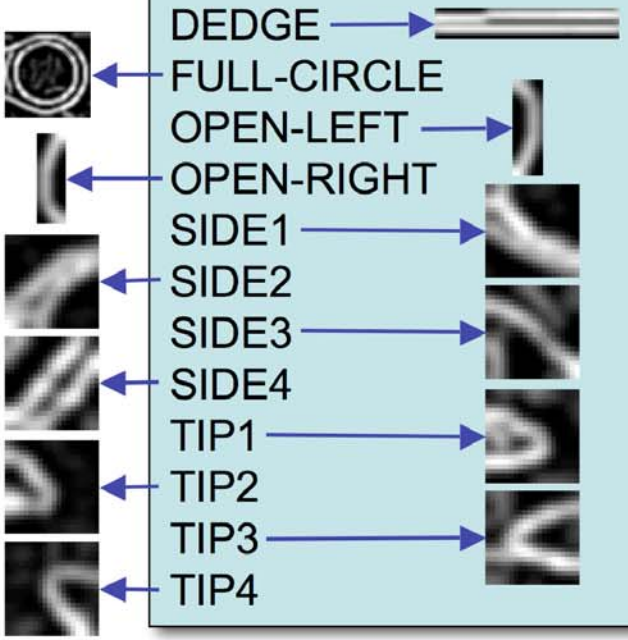
expected correlation response over sampled patches

Example Learned Grammar

wrench:
1.0 wrench → C0001 C0002 DEDGE DEDGE

C0001:
0.9752 C0001 → OPEN-RIGHT SIDE1 SIDE2 TIP1 TIP2
0.0248 C0001 → FULL-CIRCLE

C0002:
0.0174 C0002 → FULL-CIRCLE
0.9826 C0002 → OPEN-LEFT SIDE3 SIDE4 TIP3 TIP4



Example Learned Grammar

wrench:
1.0 wrench → C0001 C0002 DEDGE DEDGE

handle

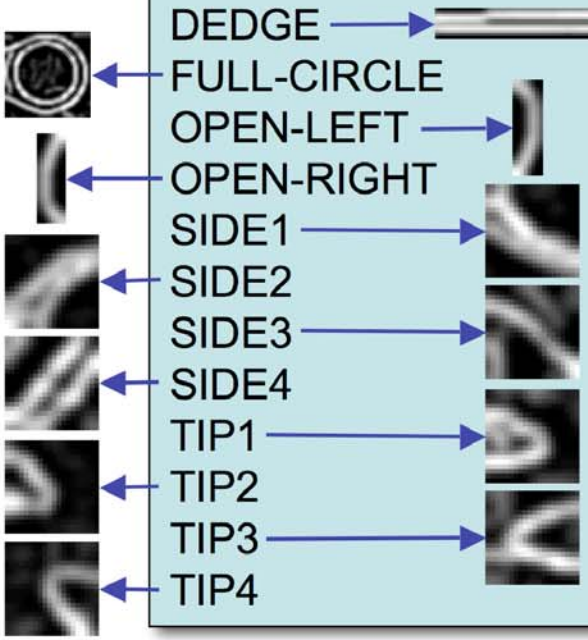
right end

C0001:
0.9752 C0001 → OPEN-RIGHT SIDE1 SIDE2 TIP1 TIP2
0.0248 C0001 → FULL-CIRCLE

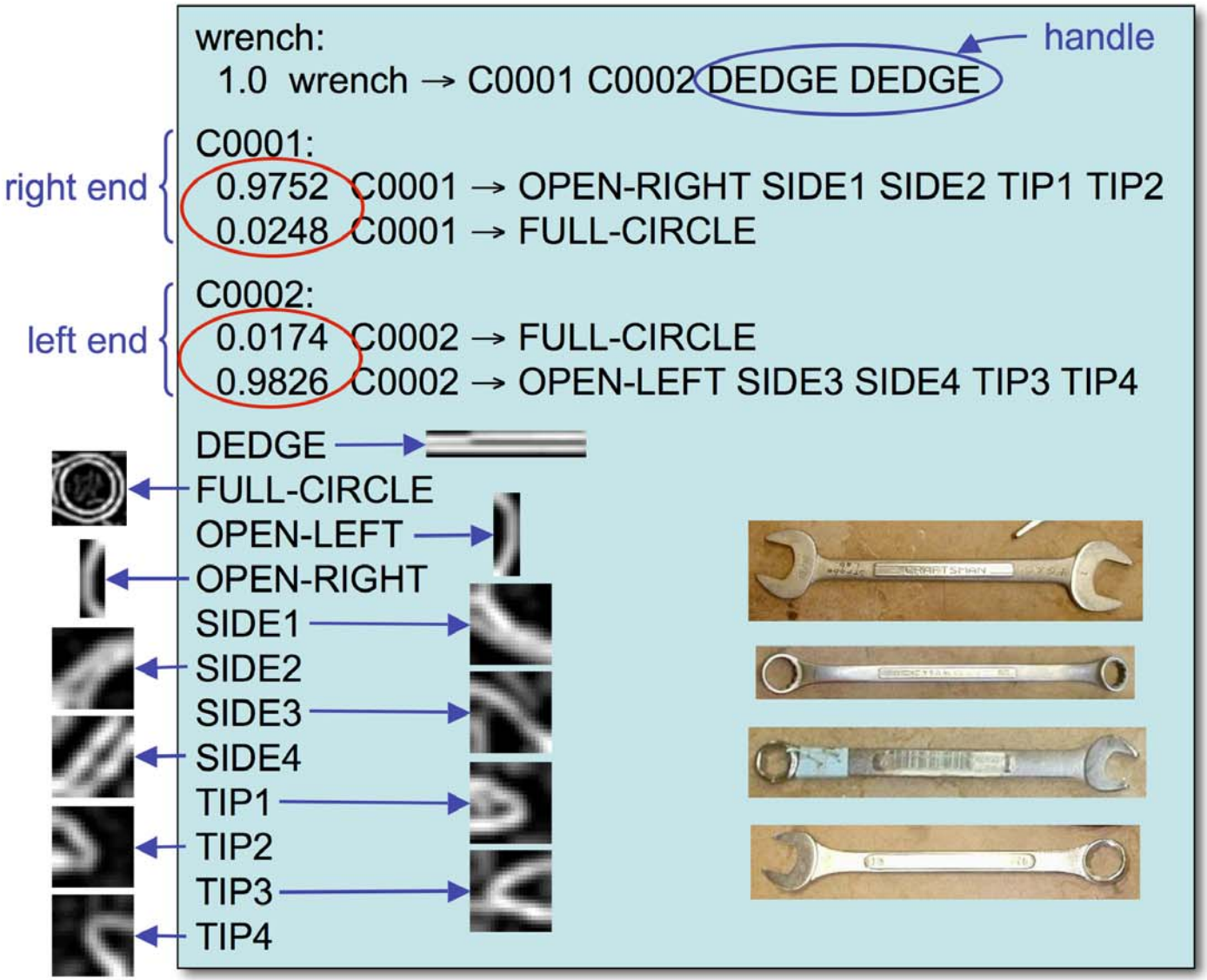
left end

C0002:
0.0174 C0002 → FULL-CIRCLE
0.9826 C0002 → OPEN-LEFT SIDE3 SIDE4 TIP3 TIP4

DEDGE
FULL-CIRCLE
OPEN-LEFT
OPEN-RIGHT
SIDE1
SIDE2
SIDE3
SIDE4
TIP1
TIP2
TIP3
TIP4

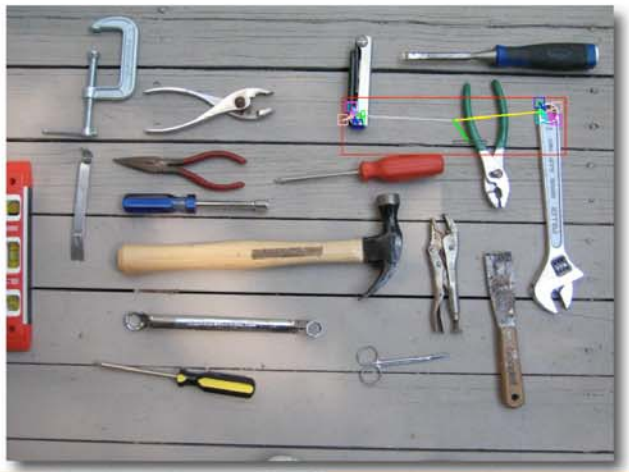


Example Learned Grammar



Example Good Detections

Example Bad Detections



Experimental Setup

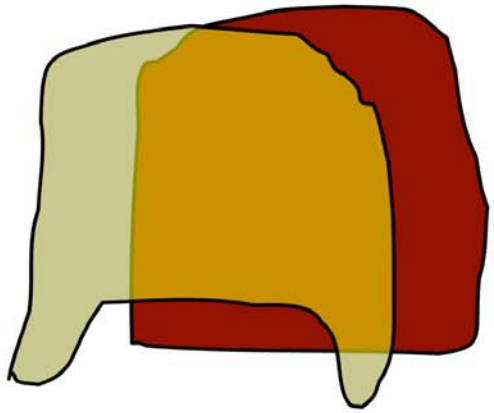
Experimental Setup

- Hypothesis: PGG enables faster learning and better generalization.
- Baseline model: set of flat models with no parameter sharing.

Experimental Setup

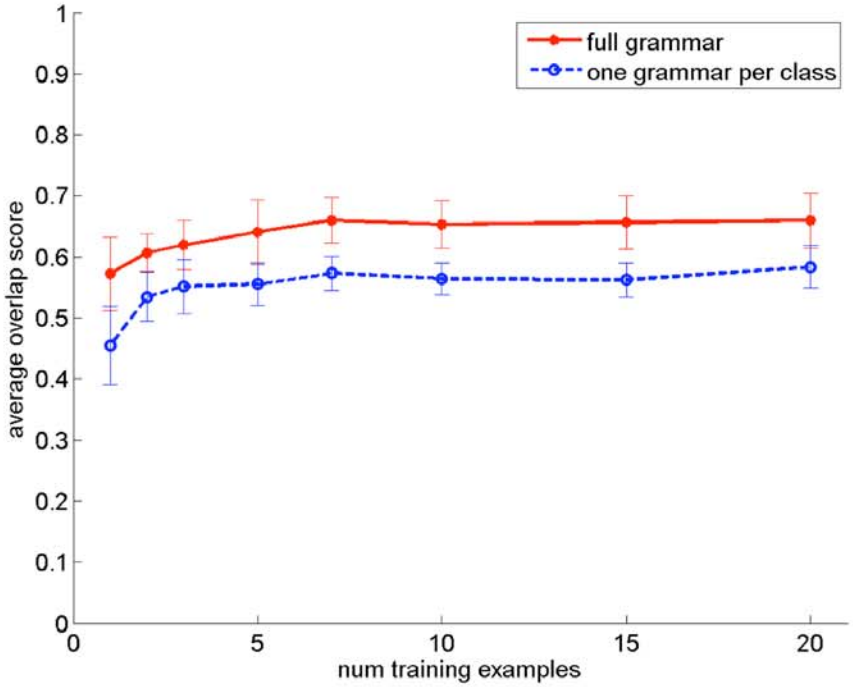
- Hypothesis: PGG enables faster learning and better generalization.
- Baseline model: set of flat models with no parameter sharing.
- Performance metric: percentage overlap between predicted and labeled bounding boxes.

$$\frac{\left|\bigcap\left(w_p, w_l\right)\right|}{\left|\bigcup\left(w_p, w_l\right)\right|}$$

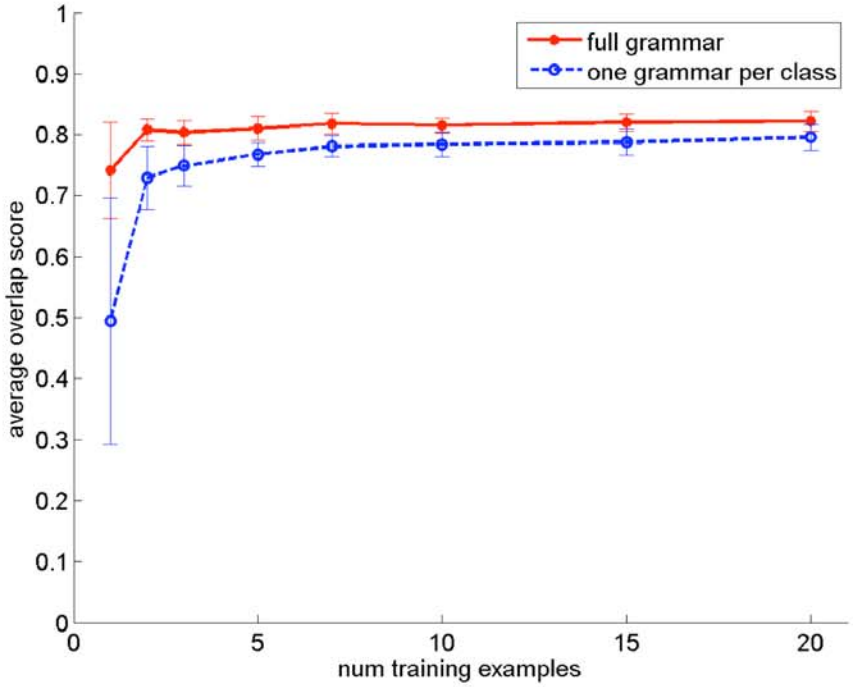


  - One when regions are identical.
  - Zero when regions do not overlap.

Experimental Results

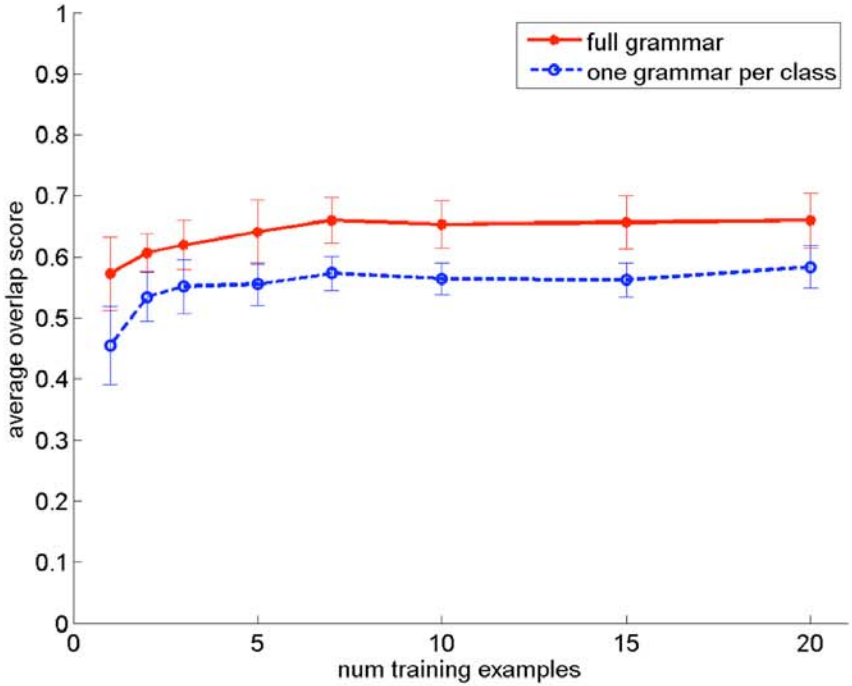


mean scores over test sets

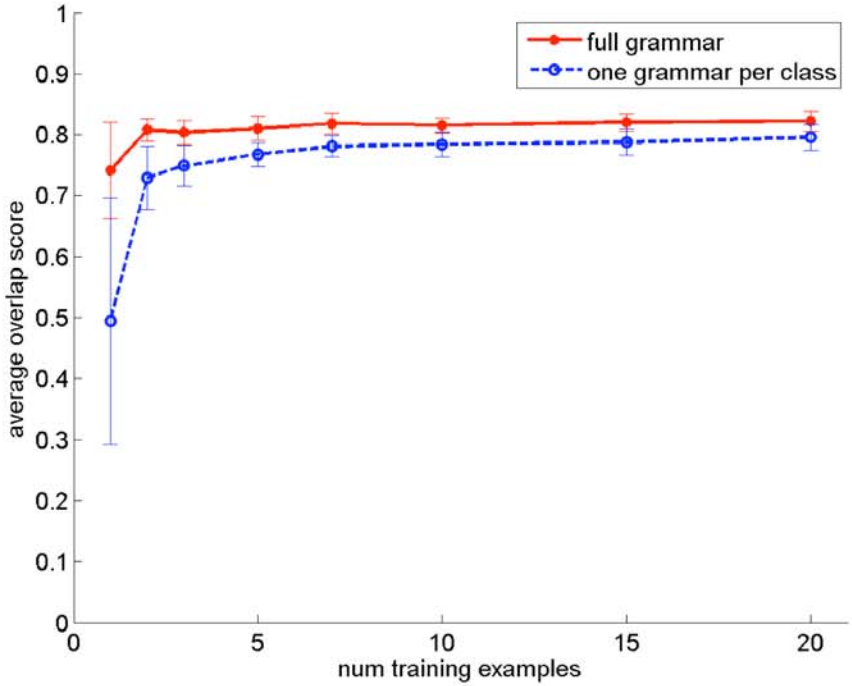


median scores over test sets

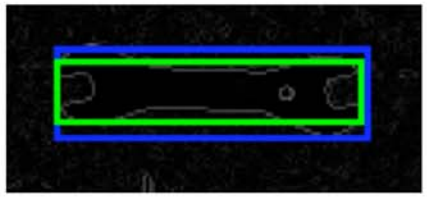
Experimental Results



mean scores over test sets



median scores over test sets



example of 0.68 overlap

Related Work

- Parts-based object class models:
  - M.A. Fischler, R.A. Elschlager (1973)
  - R. Fergus, P. Perona, A. Zisserman (2003, 2005)
  - P. Felzenszwalb, D. Huttenlocher (2005); D. Crandall et al. (2005, 2006, 2007)
- Shared or hierarchical part models:
  - A. Torralba, K. Murphy, W.T. Freeman (2004, 2005, 2007)
  - E. Sudderth, A. Torralba, W.T. Freeman, A. Willsky (2005, 2006)
  - L. Zhu, A. Yuille (2005)
  - S. Ullman, B. Epshtein (2006)
- Grammars and parsing for vision:
  - J.M. Siskind, J. Sherman, Jr., I. Pollak, M. Harper, C.A. Bouman (2003, 2007)
  - Z.W. Tu, X.R. Chen, A.L. Yuille, S.C. Zhu (2003, 2005)
  - F. Han, S.C. Zhu (2005); Z. Tu, S.C. Zhu (2006); S.C. Zhu, D. Mumford (2007); J. Porway, Z.Y. Yao, S.C. Zhu (2007); Z.T. Xu, L. Lin, S.C. Zhu (2007)
  - L. Zhu, Y. Chen, A. Yuille (2006)
  - P. Felzenszwalb, D. McAllester (2007); P. Felzenszwalb, J. Schwartz (2007)

Thank you!

Extra Slides

Object Category Recognition

Object Category Recognition

- Classification / detection:
  "Is there a chair in this image?"
- Localization:
  "Where is the chair in this image?"
- General recognition:
  "What am I looking at?"

Object Category Recognition

- Classification / detection:
  "Is there a chair in this image?"
- Localization:
  "Where is the chair in this image?"
- General recognition:
  "What am I looking at?"
- Learning
  - Appearance: "What do chair parts look like?"
  - Parts / Structure: "How is a chair different than a stool?"
  - Classes: "What types of objects exist in the world?"

Object Category Recognition

- **Classification / detection:**
  "Is there a chair in this image?"
- **Localization:**
  "Where is the chair in this image?"
- General recognition:
  "What am I looking at?"
- Learning
  - Appearance: "What do chair parts look like?"
  - **Parts / Structure: "How is a chair different than a stool?"**
  - Classes: "What types of objects exist in the world?"

Motivation

(Figure from Blanz et al., 1996.)

Motivation

- Human-made objects: constrained by function and custom.

(Figure from Blanz et al., 1996.)

Motivation

- Human-made objects: constrained by function and custom.
- High variability:
  - Shape and appearance.
  - Presence and number of object parts.

(Figure from Blanz et al., 1996.)

Motivation

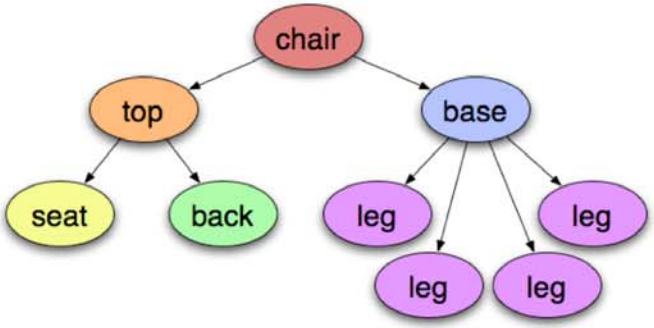
- Human-made objects: constrained by function and custom.
- High variability:
  - Shape and appearance.
  - Presence and number of object parts.
- Shared parts and structures within and among related classes.

(Figure from Blanz et al., 1996.)

Approach

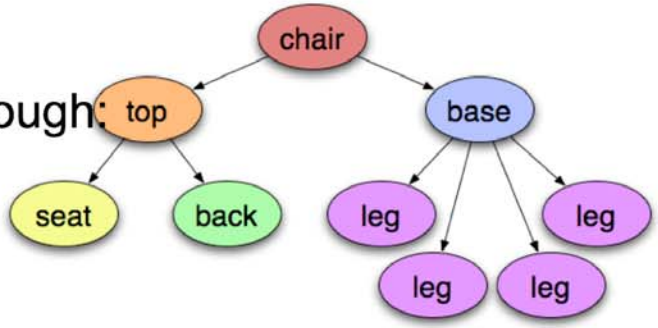
Approach

- Probabilistic geometric grammars (PGGs): extend probabilistic context-free grammars (PCFGs) with geometric and appearance models.



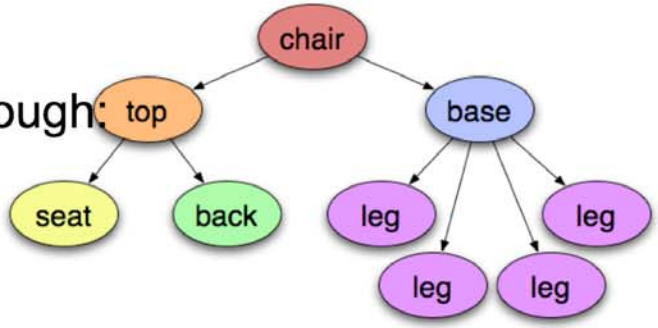
Approach

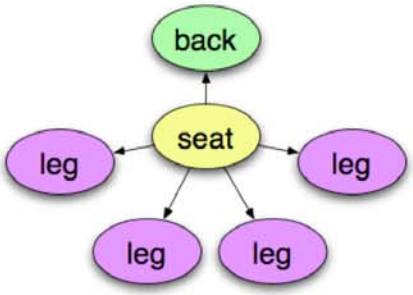
- Probabilistic geometric grammars (PGGs): extend probabilistic context-free grammars (PCFGs) with geometric and appearance models.
- Faster learning and better generalization through:
  - Conditional independence assumptions.



Approach

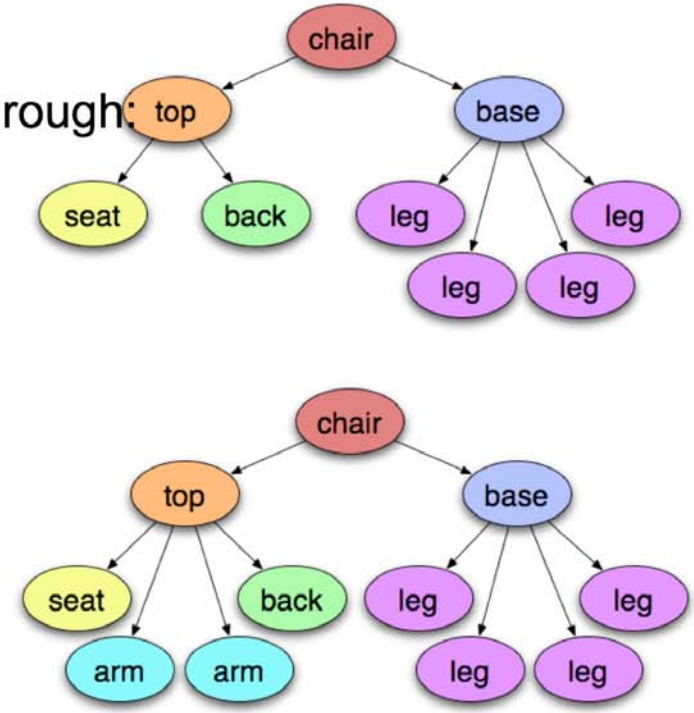
- Probabilistic geometric grammars (PGGs): extend probabilistic context-free grammars (PCFGs) with geometric and appearance models.
- Faster learning and better generalization through:
  - Conditional independence assumptions.





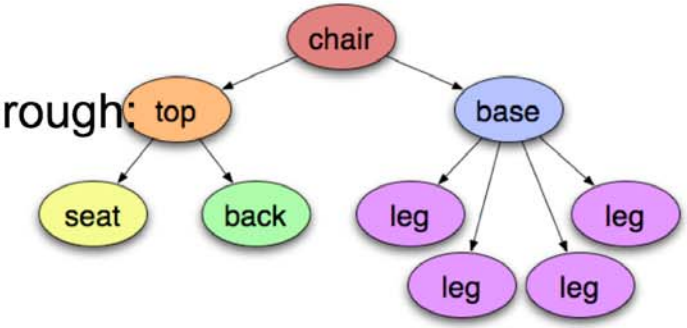
Approach

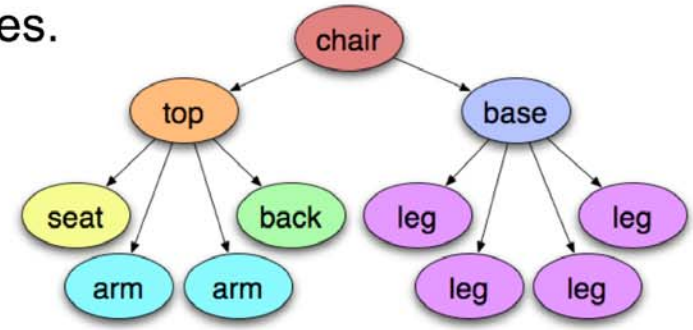
- Probabilistic geometric grammars (PGGs): extend probabilistic context-free grammars (PCFGs) with geometric and appearance models.
- Faster learning and better generalization through:
  - Conditional independence assumptions.
  - Sharing of parts and structure among classes.



Approach

- Probabilistic geometric grammars (PGGs): extend probabilistic context-free grammars (PCFGs) with geometric and appearance models.
- Faster learning and better generalization through:
  - Conditional independence assumptions.
  - Sharing of parts and structure among classes.
- Search simultaneously for shared substructures.

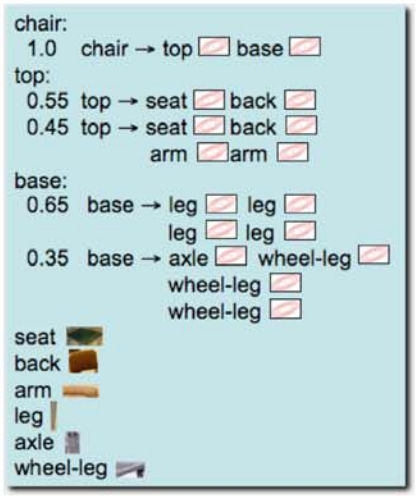




Components

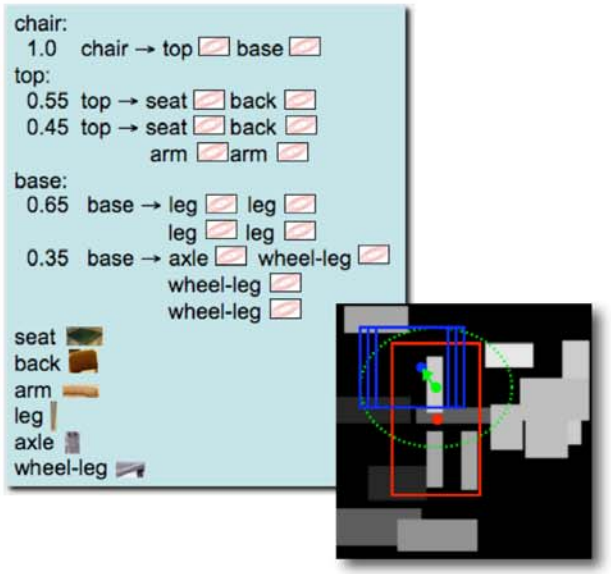
Components

- Model: probabilistic geometric grammars.



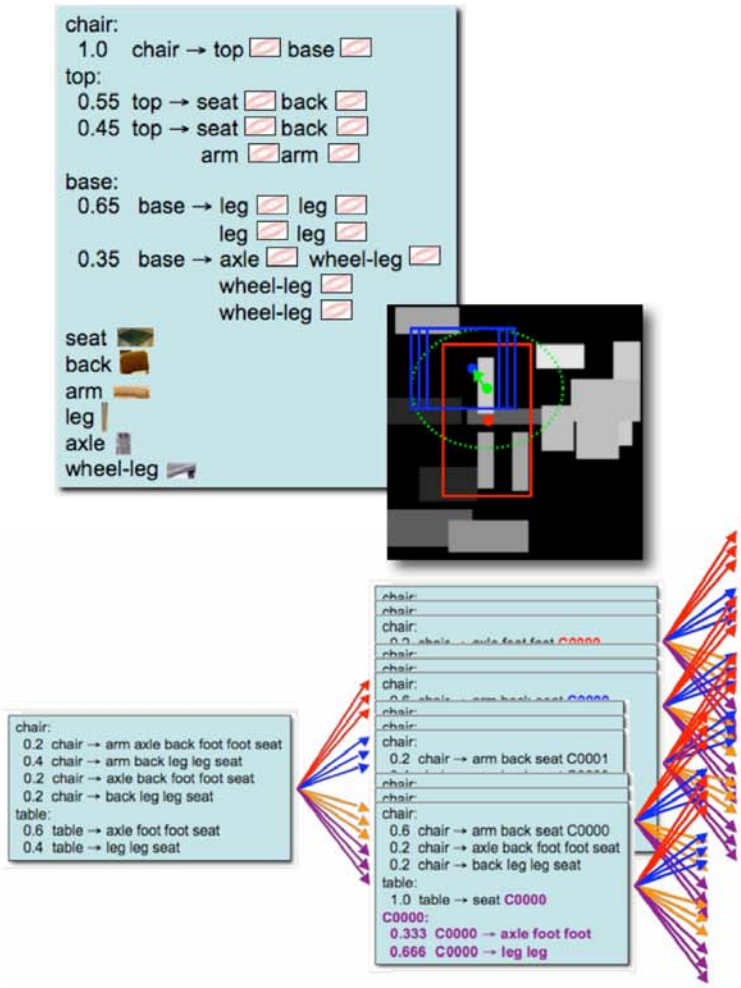
Components

- Model: probabilistic geometric grammars.
- Recognition: dynamic programming parsing algorithm to find best object class or location in image.



Components

- Model: probabilistic geometric grammars.
- Recognition: dynamic programming parsing algorithm to find best object class or location in image.
- Learning:
  - Parameters: estimate from data using EM.
  - Structure: incremental search through space of grammars.



Efficient Parsing with PGGs

Efficient Parsing with PGGs

- Cannot simply adapt algorithms for parsing strings with PCFGs.

0.30	NP → ART NP
0.25	NP → ADJ NP
0.45	NP → N
1.0	ART → the
0.45	ADJ → big
0.55	ADJ → red
0.50	N → barn
0.50	N → ball

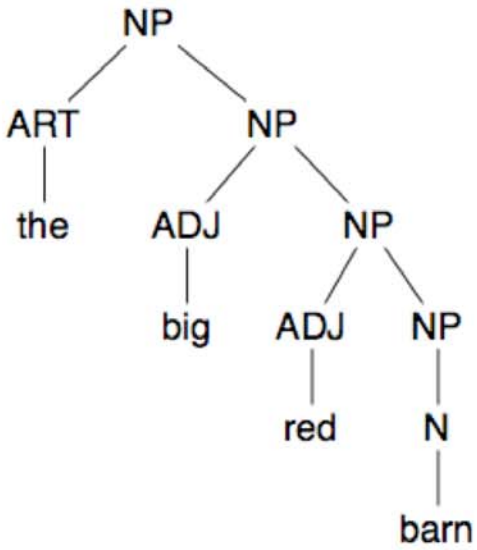
"the big red barn"

Efficient Parsing with PGGs

- Cannot simply adapt algorithms for parsing strings with PCFGs.

0.30	NP → ART NP
0.25	NP → ADJ NP
0.45	NP → N
1.0	ART → the
0.45	ADJ → big
0.55	ADJ → red
0.50	N → barn
0.50	N → ball

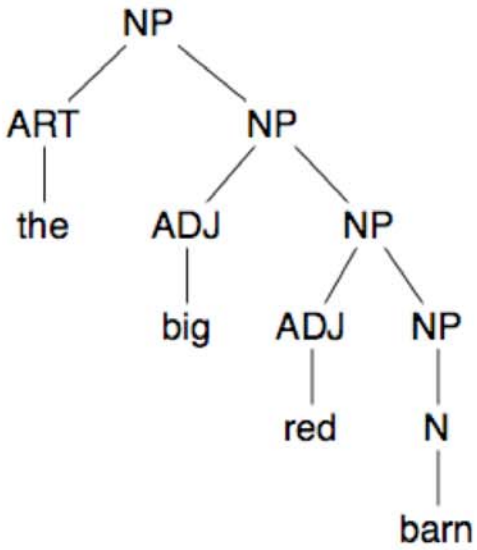
"the big red barn"



Efficient Parsing with PGGs

- Cannot simply adapt algorithms for parsing strings with PCFGs.

0.30 NP → ART NP
0.25 NP → ADJ NP
0.45 NP → N
1.0 ART → the
0.45 ADJ → big
0.55 ADJ → red
0.50 N → barn
0.50 N → ball



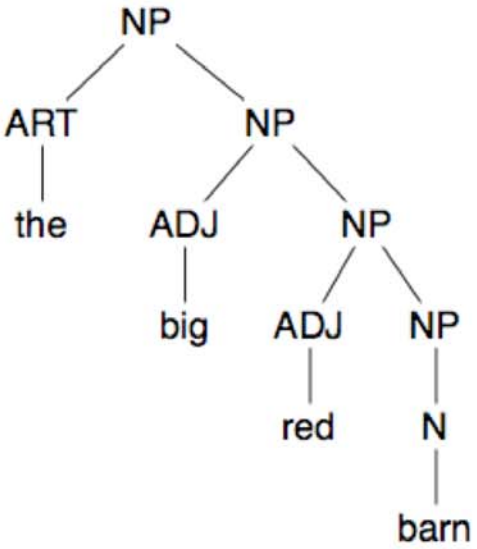
"the big red barn"

- No inherent ordering on object parts in 2D (or 3D) space.

Efficient Parsing with PGGs

- Cannot simply adapt algorithms for parsing strings with PCFGs.

0.30 NP → ART NP
0.25 NP → ADJ NP
0.45 NP → N
1.0 ART → the
0.45 ADJ → big
0.55 ADJ → red
0.50 N → barn
0.50 N → ball



"the big red barn"

- No inherent ordering on object parts in 2D (or 3D) space.
- Must deal with clutter.

Efficient Parsing with PGGs

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

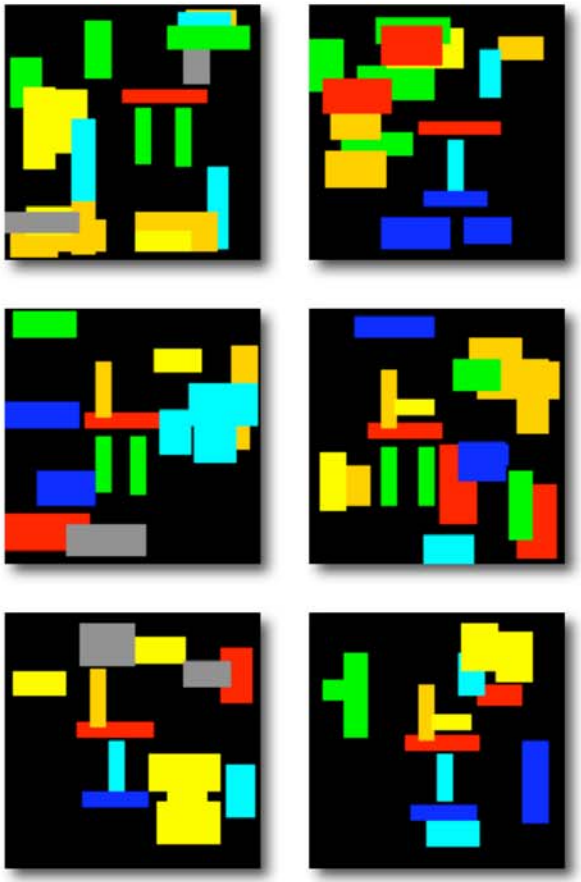
- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.

Efficient Parsing with PGGs

- Hierarchical extension of Felzenszwalb & Huttenlocher (2005).
- Depends on discretization of image into fixed set of regions.
- Dynamic programming lates score for each region and part model, caching intermediate results.

A Parsing Example

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.



A Parsing Example

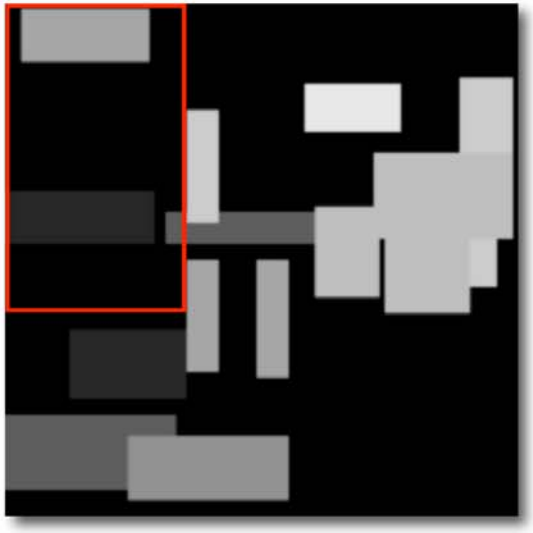
table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example

current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

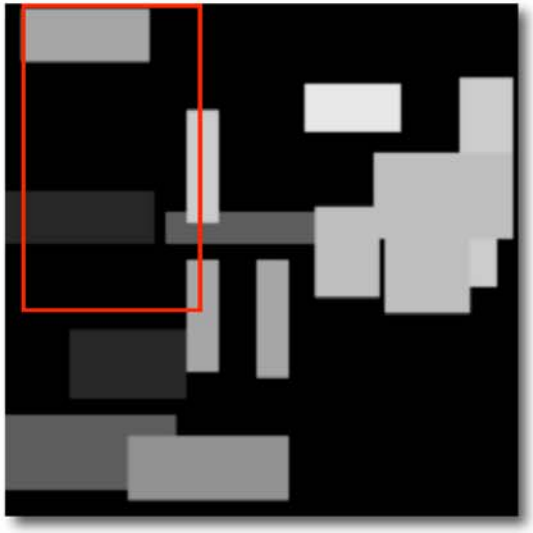
A Parsing Example



current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

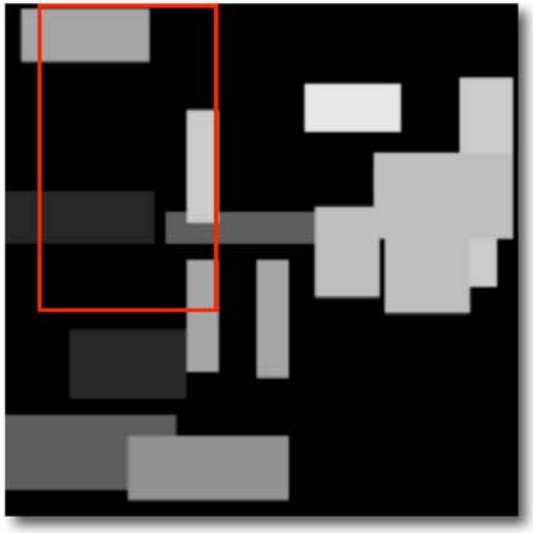
A Parsing Example



current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

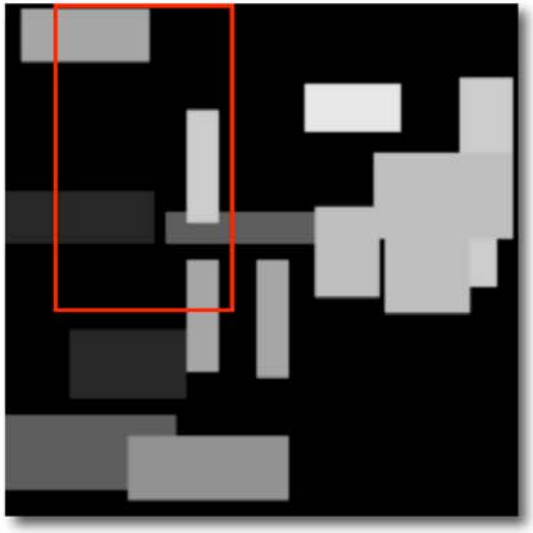
A Parsing Example



current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example

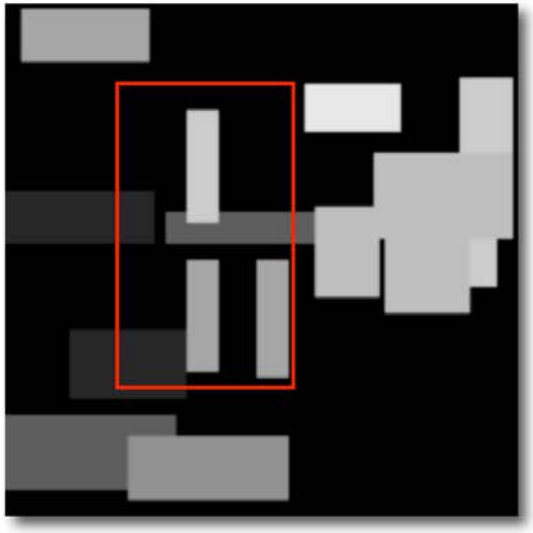
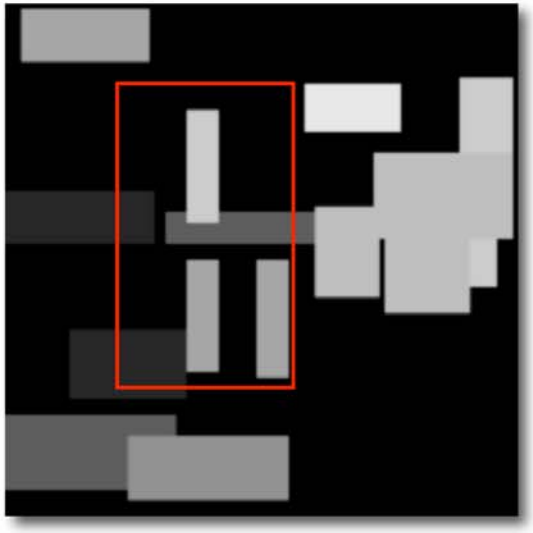


table:
1.0 table → seat, base .
chair:
1.0 chair → chair-top, base .
chair-top:
0.5 top → seat, back .
0.5 top → seat, back, arm .
base:
0.5 base → leg, leg .
0.5 base → axle, foot, foot .
seat .
back .
arm .
leg .
axle .
foot .

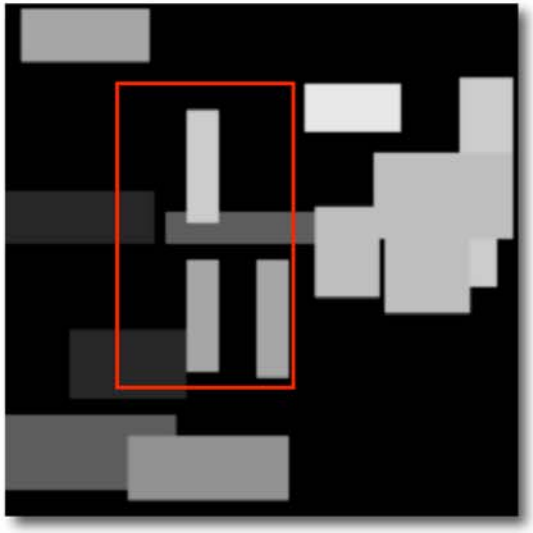
A Parsing Example



current part model: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



current part model: chair

current rule: chair → chair-top, base.

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example

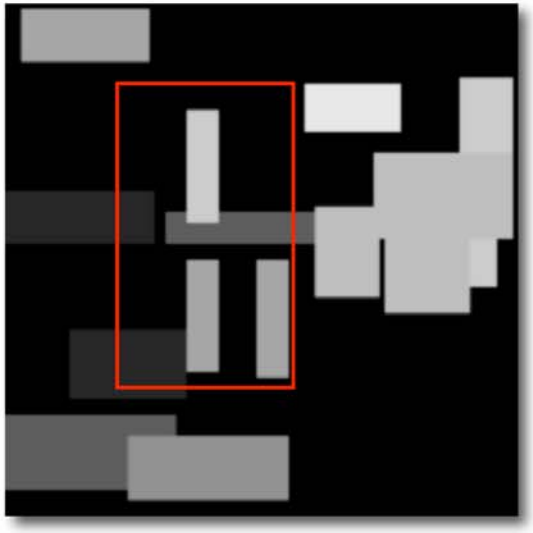


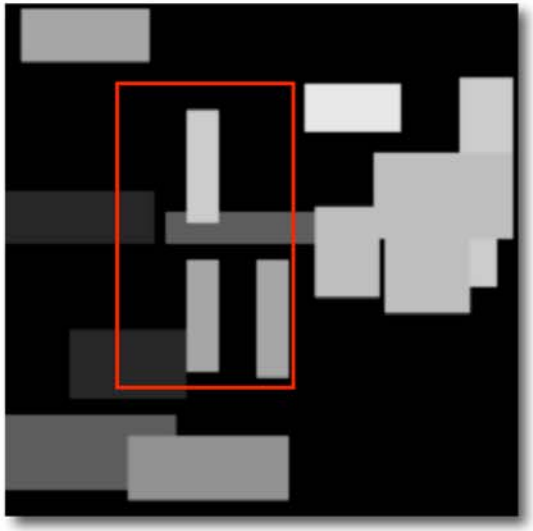
table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

sum over rules

current part model: chair

current rule: chair → chair-top, base.

A Parsing Example



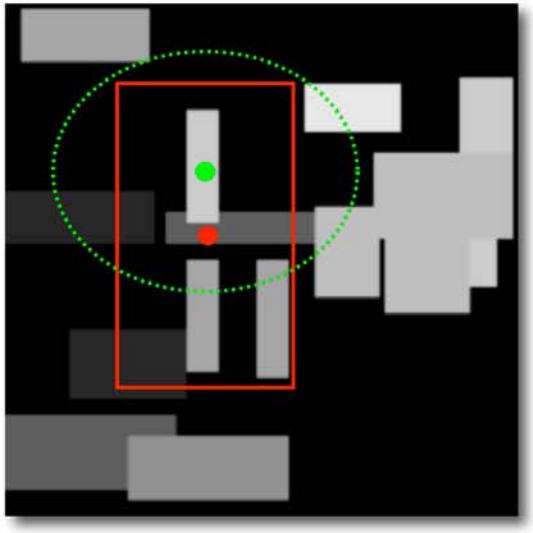
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



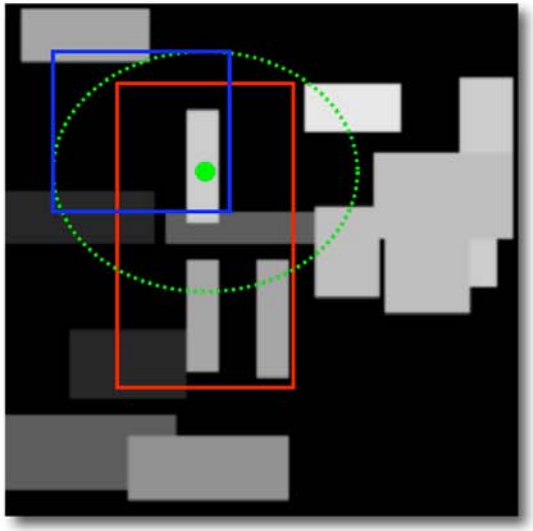
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



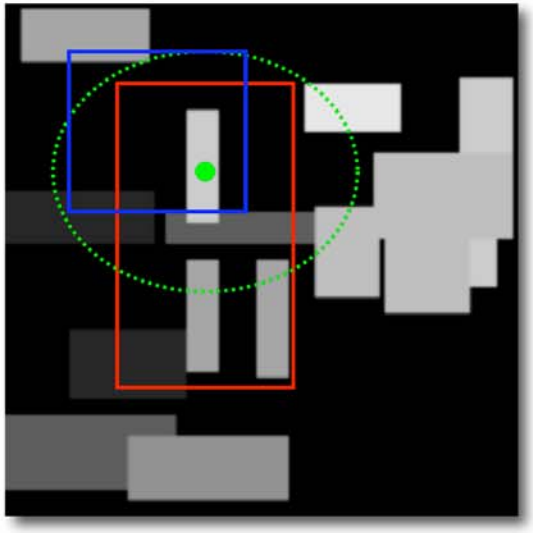
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



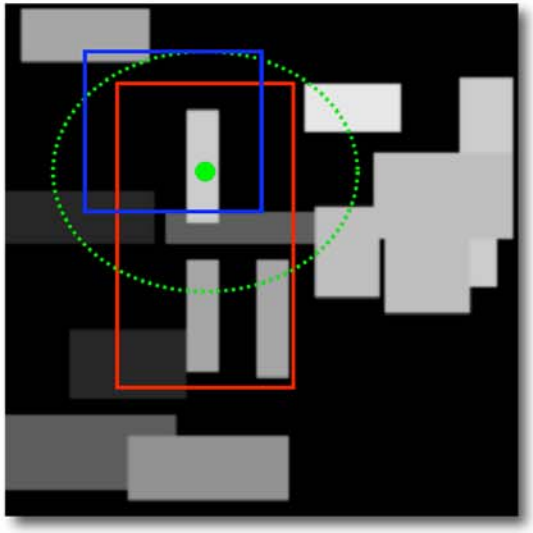
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



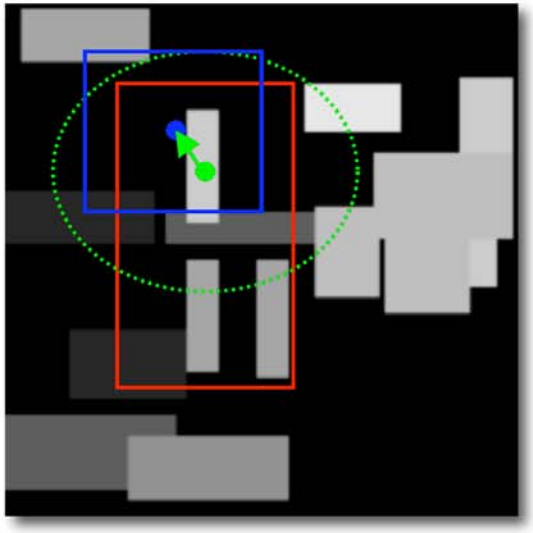
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



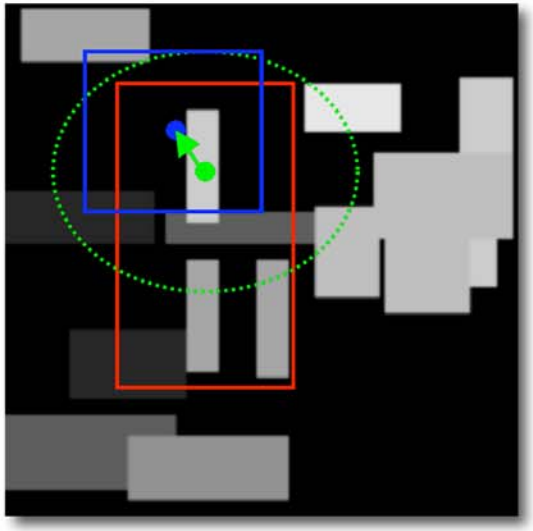
current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example

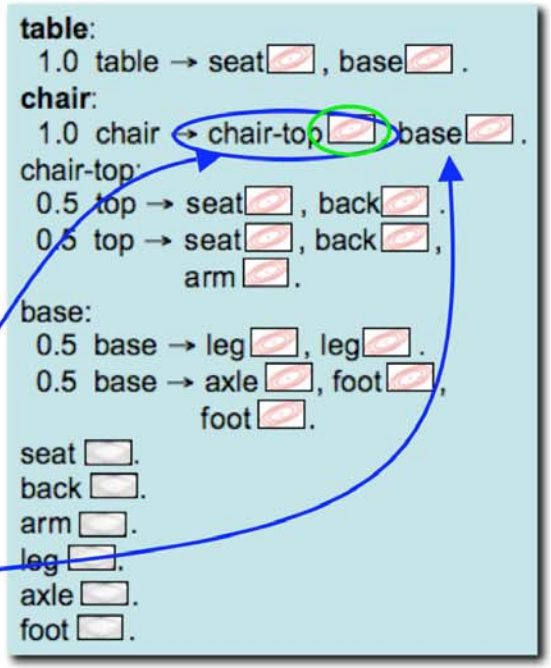


current part model: chair

current rule: chair → chair-top, base.

current subpart: chair-top

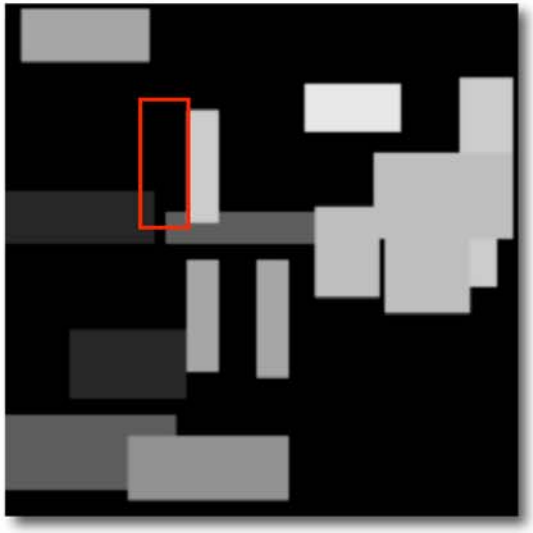
product of subpart values, weighted by geometry



A Parsing Example

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



current part model: back

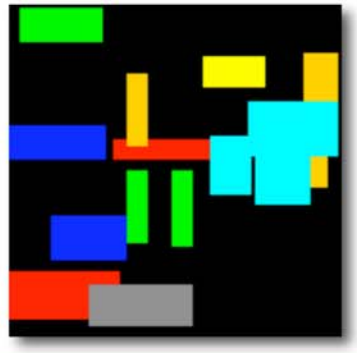
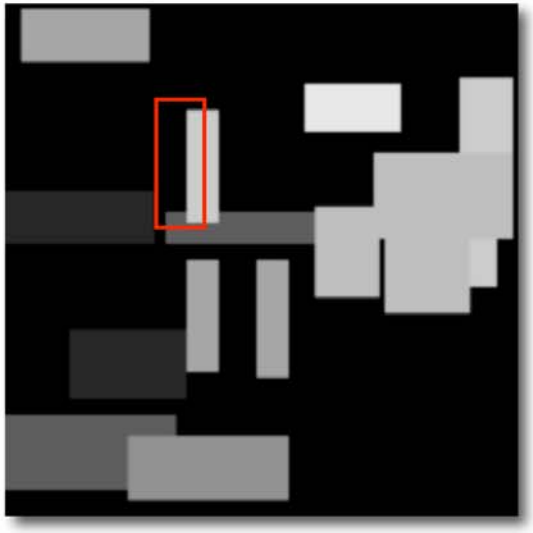


table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

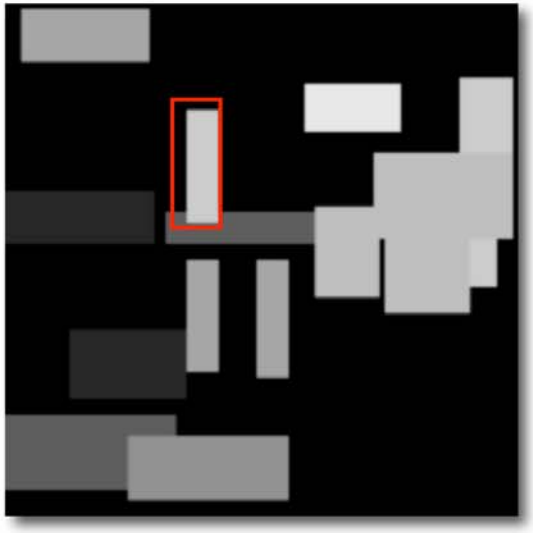
A Parsing Example



current part model: back

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

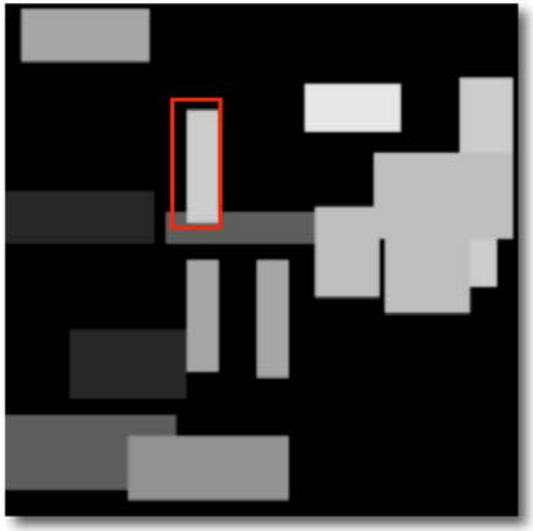
A Parsing Example



current part model: back

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example



current part model: back

evaluate appearance model

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

A Parsing Example

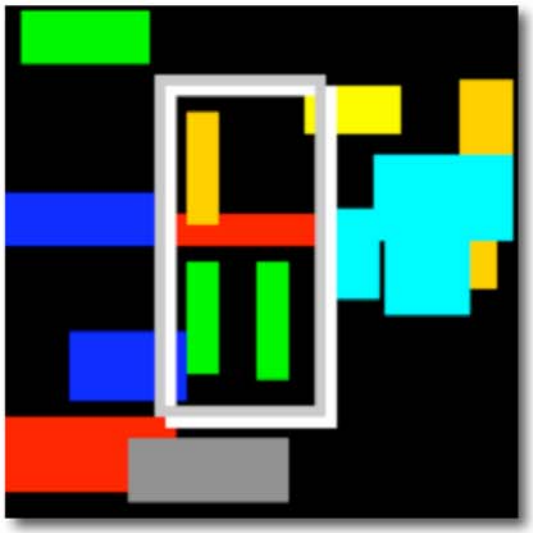
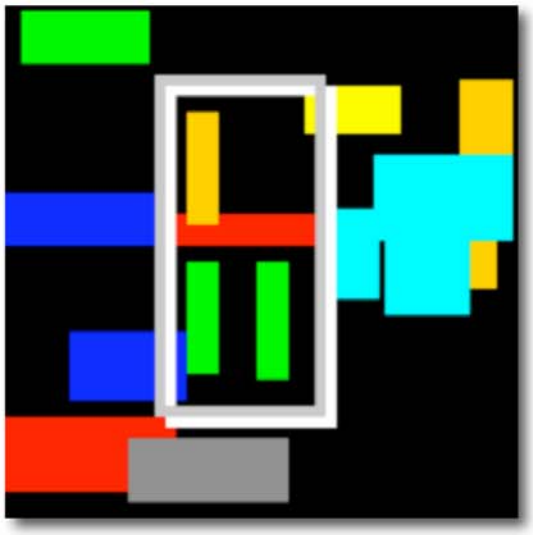


table:
1.0 table → seat, base .
chair:
1.0 chair → chair-top, base .
chair-top:
0.5 top → seat, back .
0.5 top → seat, back, arm .
base:
0.5 base → leg, leg .
0.5 base → axle, foot, foot .
seat .
back .
arm .
leg .
axle .
foot .

A Parsing Example



predicted object class: chair

table:
1.0 table → seat, base.
chair:
1.0 chair → chair-top, base.
chair-top:
0.5 top → seat, back.
0.5 top → seat, back, arm.
base:
0.5 base → leg, leg.
0.5 base → axle, foot, foot.
seat.
back.
arm.
leg.
axle.
foot.

Parameter Learning in PGGs

Parameter Learning in PGGs

- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Parameter Learning in PGGs

- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:

$$P(cw \mid I^i, \rho u) \qquad P(crw \mid I^i, \rho u) \qquad P(crw, d_{crk} v \mid I^i, \rho u)$$

Parameter Learning in PGGs

- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:

training image i

$$P(cw \mid I^i, \rho u) \qquad P(crw \mid I^i, \rho u) \qquad P(crw, d_{crk} v \mid I^i, \rho u)$$

Parameter Learning in PGGs

- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:

training image i

root class ρ occupies labeled window u

$$P(cw \mid I^i, \rho u) \qquad P(crw \mid I^i, \rho u) \qquad P(crw, d_{crk} v \mid I^i, \rho u)$$

Parameter Learning in PGGs

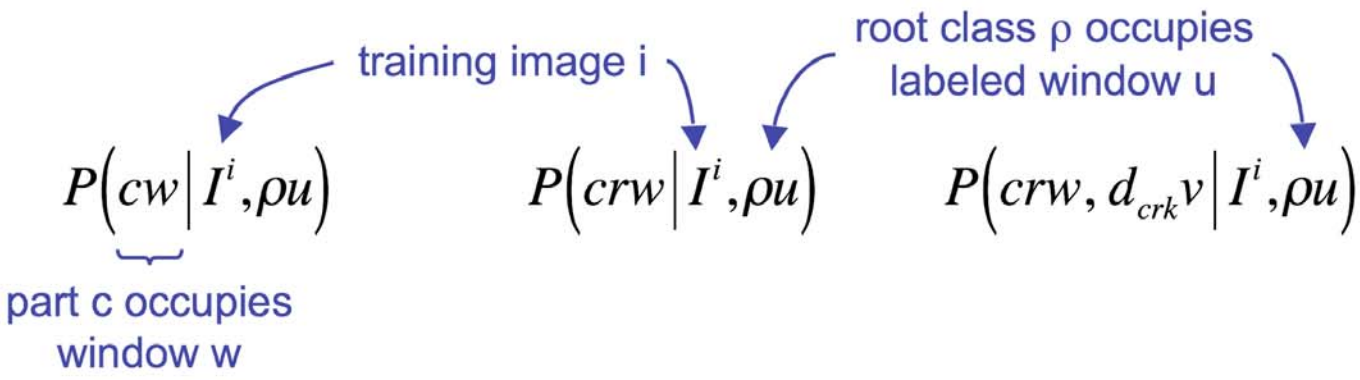
- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:



Parameter Learning in PGGs

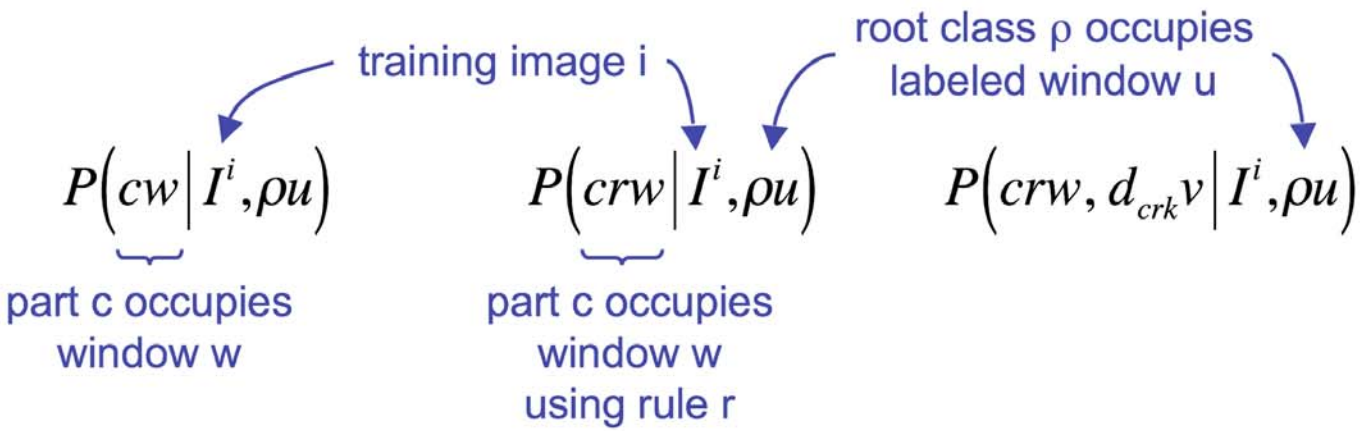
- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:



Parameter Learning in PGGs

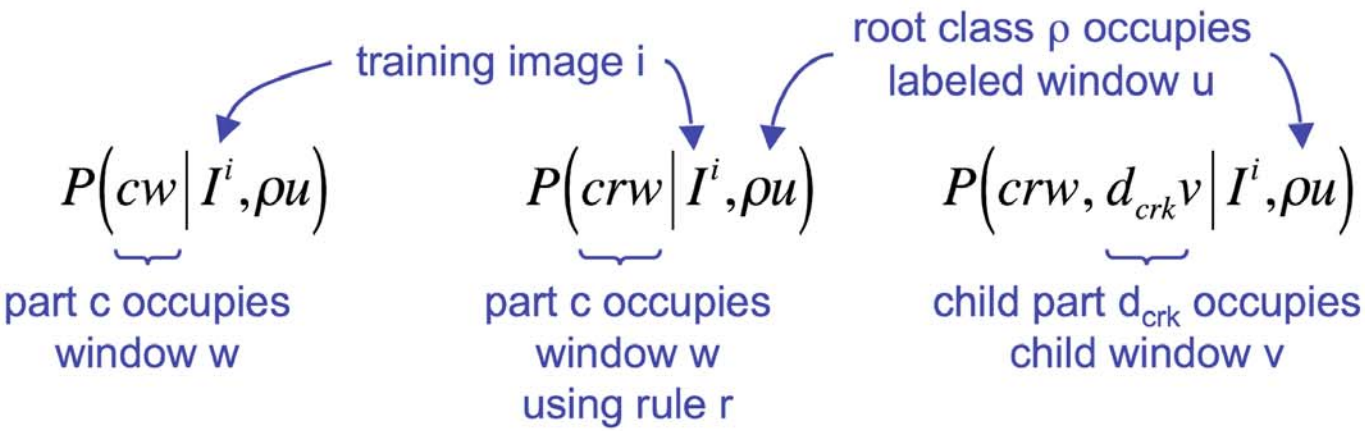
- **E-step:**

$$\beta(c,w) = \frac{P(I_w \mid cw)}{P(I_w)} \qquad \alpha(c,w) = \frac{P(I_{\bar{w}} \mid cw)}{P(I_{\bar{w}})}$$

image likelihood ratio for pixels *inside* w

image likelihood ratio for pixels *outside* w

Calculate responsibilities for each training image:



Parameter Learning in PGGs

- **M-step:** Reestimate parameters from data, weighted by responsibilities

Parameter Learning in PGGs

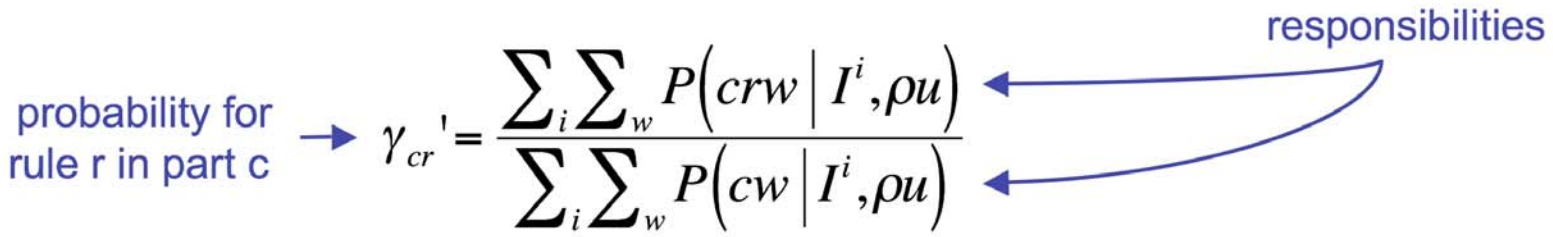
- **M-step:** Reestimate parameters from data, weighted by responsibilities

probability for rule r in part c →

$$\gamma_{cr}' = \frac{\sum_i \sum_w P\left(crw \mid I^i, \rho u\right)}{\sum_i \sum_w P\left(cw \mid I^i, \rho u\right)}$$

Parameter Learning in PGGs

- **M-step:** Reestimate parameters from data, weighted by responsibilities



Parameter Learning in PGGs

- **M-step:** Reestimate parameters from data, weighted by responsibilities

probability for rule r in part c

$$\gamma_{cr}' = \frac{\sum_i \sum_w P\left(crw \mid I^i, \rho u\right)}{\sum_i \sum_w P\left(cw \mid I^i, \rho u\right)}$$

responsibilities

mean for rule part k in rule r in part c

$$\mu_{crk}' = \frac{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right) T(v, w)}{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right)}$$

Parameter Learning in PGGs

- **M-step:** Reestimate parameters from data, weighted by responsibilities

probability for rule r in part c

$$\gamma_{cr}' = \frac{\sum_i \sum_w P\left(crw \mid I^i, \rho u\right)}{\sum_i \sum_w P\left(cw \mid I^i, \rho u\right)}$$

responsibilities

mean for rule part k in rule r in part c

$$\mu_{crk}' = \frac{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right) T(v, w)}{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right)}$$

child window transformed by parent

Parameter Learning in PGGs

- **M-step:** Reestimate parameters from data, weighted by responsibilities

probability for rule r in part c

$$\gamma_{cr}' = \frac{\sum_i \sum_w P\left(crw \mid I^i, \rho u\right)}{\sum_i \sum_w P\left(cw \mid I^i, \rho u\right)}$$

responsibilities

mean for rule part k in rule r in part c

$$\mu_{crk}' = \frac{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right) T(v,w)}{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right)}$$

child window transformed by parent

covariance for rule part k in rule r in part c

$$\Sigma_{crk}' = \frac{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right) \left(T(v,w) - \mu_{crk}'\right)^2}{\sum_i \sum_w \sum_v P\left(crw, d_{crk} v \mid I^i, \rho u\right)}$$

Search Control

Search Control

- Combine parts with closest geometry models according to symmetrized KL divergence.

Search Control

- Combine parts with closest geometry models according to symmetrized KL divergence.
- Apply existing AND and OR parts before creating new parts.

Search Control

- Combine parts with closest geometry models according to symmetrized KL divergence.
- Apply existing AND and OR parts before creating new parts.
- Prune proposals with high variance in geometry models.

Future Work

- Address multimodal data geometry:
  - Mixtures of Gaussians.
  - Automatically learn separate part models or rules for different mixture components.
- Improve appearance models:
  - Richer shape models for regions.
  - Detectors / models for corners, interest points, or texture.
  - Lower notion of primitive part model to have a set of image feature models.
  - Discriminative approaches.
- Learn grammar structure:
  - Build hierarchies to minimize variance of geometric and appearance models.
  - Minimize complexity of hierarchy to maximize shared structure.
- Reincorporate 3D models into 2D PGG:
  - 2D recognition step.
  - 3D reconstruction phase, dependent on both image data and probabilistic 3D object class model.