Bayesian Grammar Learning for Inverse Procedural Modeling

June 2013 | CVPR

Anđelo Martinović[1], Luc Van Gool[1,2]

[1]ESAT/PSI/VISICS @ KU Leuven
[2]Computer Vision Lab @ ETH Zurich

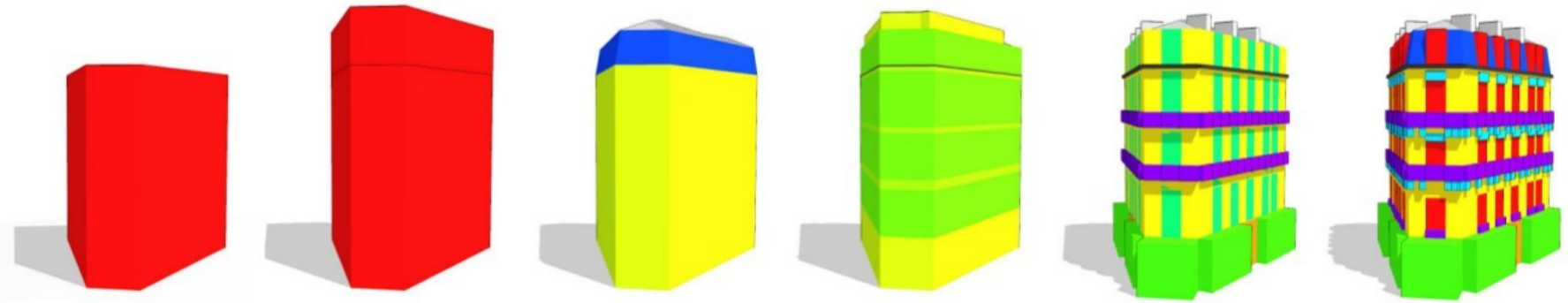
KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Procedural modeling



KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Procedural modeling – Esri CityEngine

Procedural modeling – Esri CityEngine

Procedural modeling – Esri CityEngine

Procedural modeling – Esri CityEngine

Inverse procedural modeling


```
Lot      --> extrude(height) Building
Building --> comp(f) { side: Facade | top: Roof }
Facade   --> split(y) {'0.2: Groundfloor | {'0.01:
Ledge | '0.29: Floor}* }
Floor    --> split(x) {
       '0.05: Wall
     | {'0.01: Wall| '0.08: WindowTile | '0.01:
Wall}*
     | '0.05: Wall}

   ...
```

Related work

Rules are known in advance

- Metropolis Procedural Modeling (Talton et al. 2011)

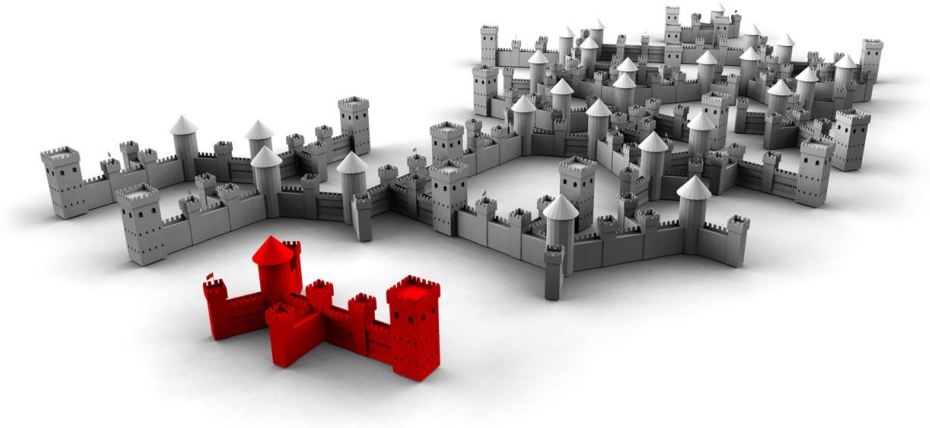
- Random Exploration of the Procedural Space for Single-View 3D Modeling of Buildings (Simon et al. 2009)

  Shape Grammar Parsing via Reinforcement Learning (Teboul et al. 2011)

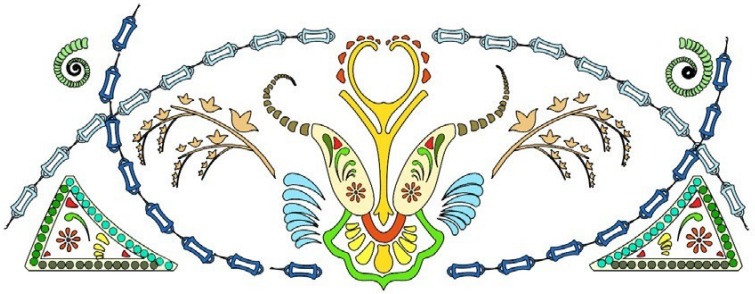
Related work

Rules are learned

- A Connection between Partial Symmetry and Inverse Procedural Modeling (Bokeloh et al. 2010)



- Inverse procedural modeling by automatic generation of L-systems (Šťava et al. 2010)



KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Related work

Related work

- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns

Related work

- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns

Related work

- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns

Related work

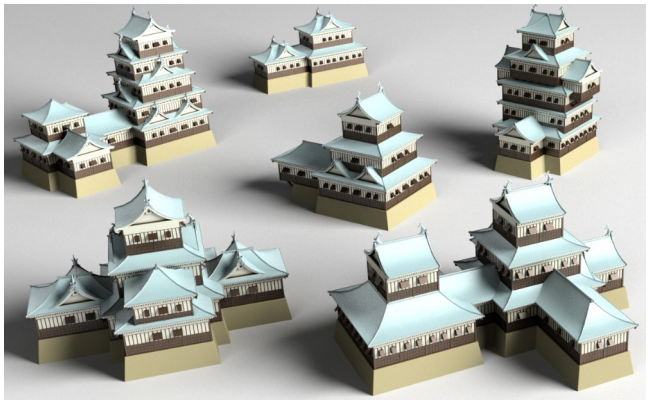
- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns

Related work

- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns

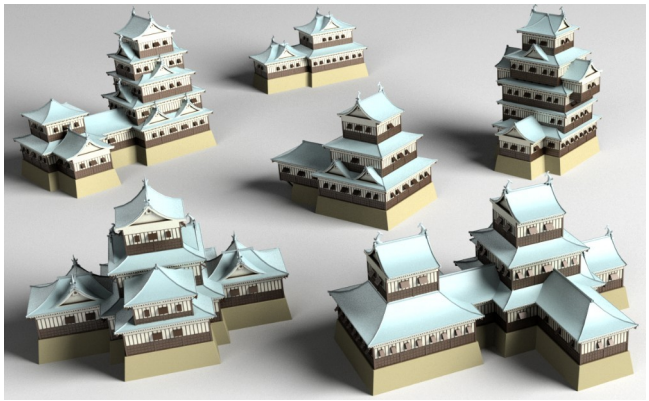
Related work

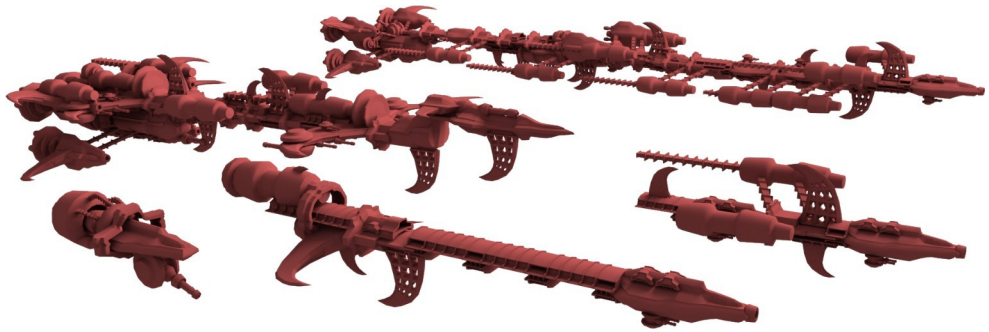
- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns



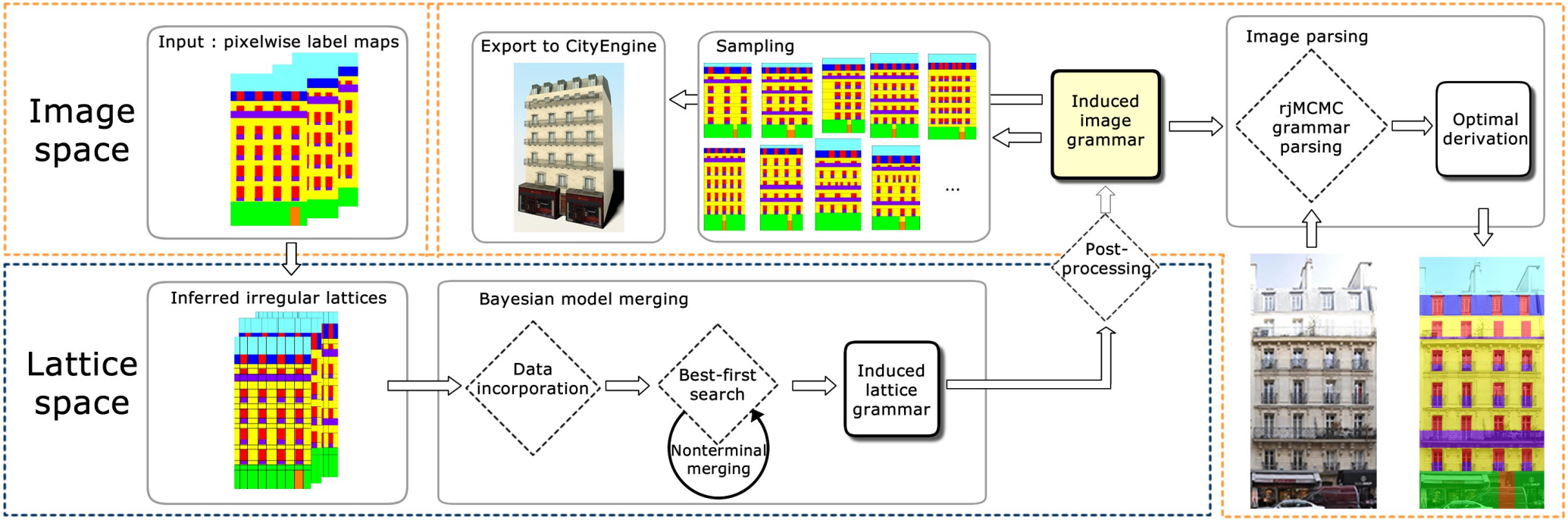
Related work

- Grammar learning
  - Gold, 1967: *No superfinite family of deterministic languages (including regular and context-free languages) can be identified in the limit*
  - Horning, 1969: *Stochastic* context free grammars *can* be learned from only positive examples
  - Stolcke, 1994: Introducing **Bayesian Model Merging (BMM)** for HMM, SCFG and PAG induction
  - Hwang et al, 2011: Rediscovering BMM for induction of probabilistic programs
  - Talton et al, 2012: BMM is used to learn design patterns



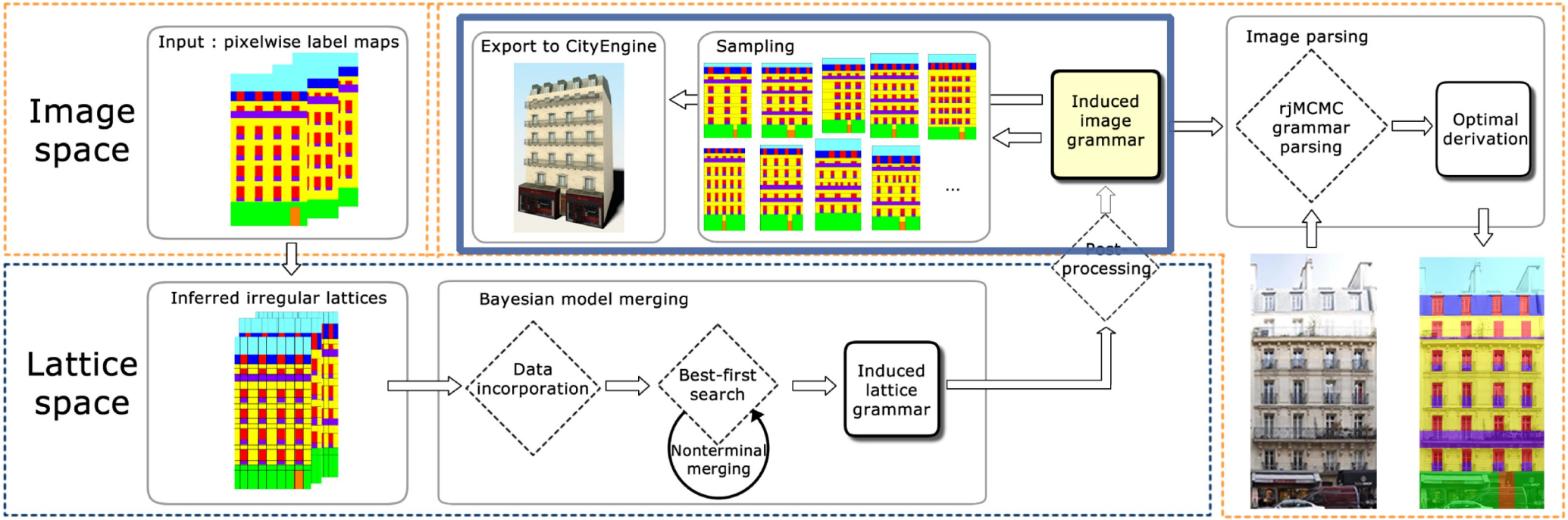


Method overview



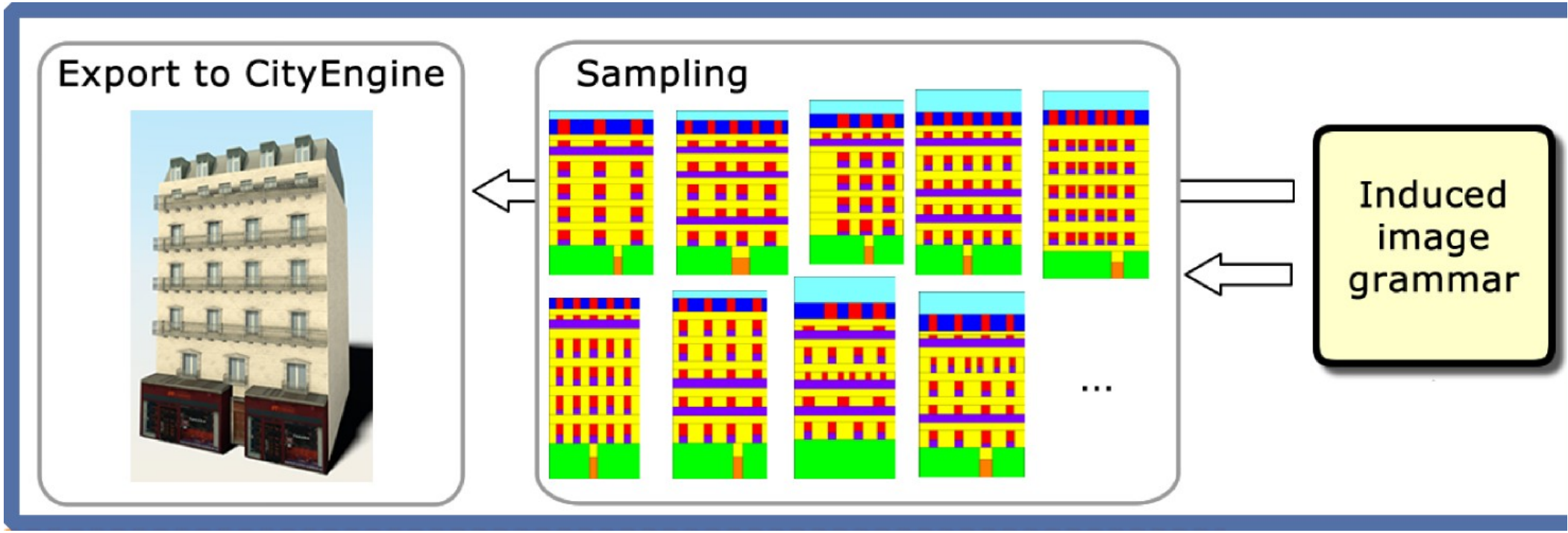
How can we use the learned grammar?

- Sampling novel buildings



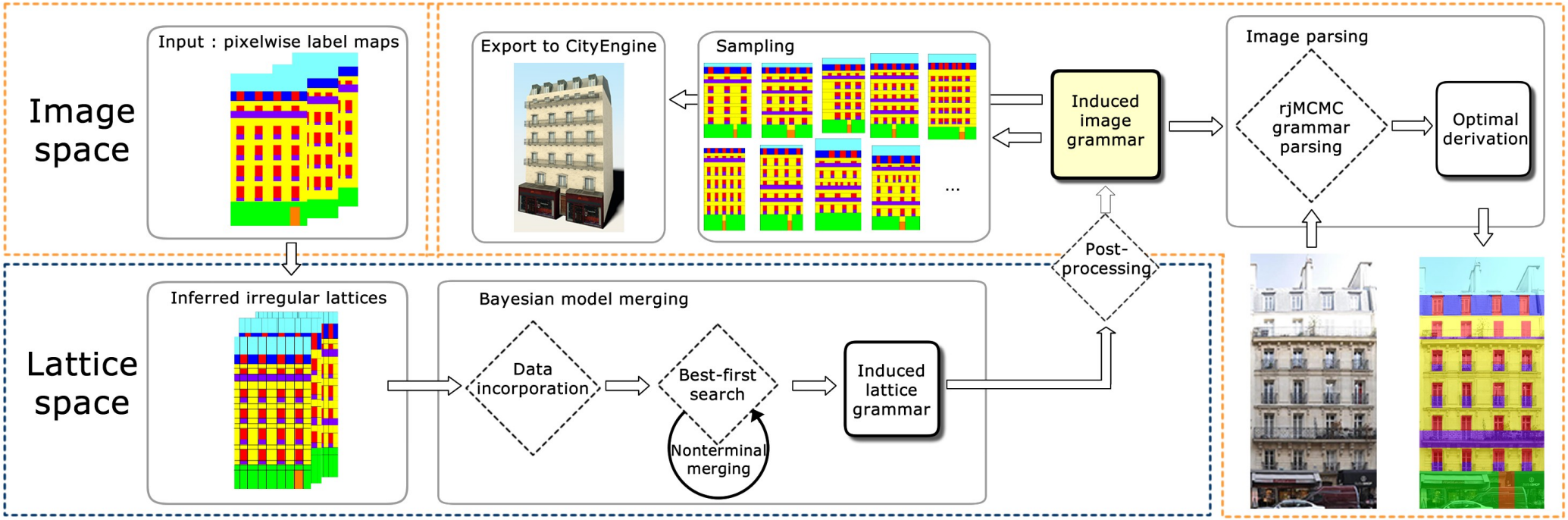
How can we use the learned grammar?

- Sampling novel buildings



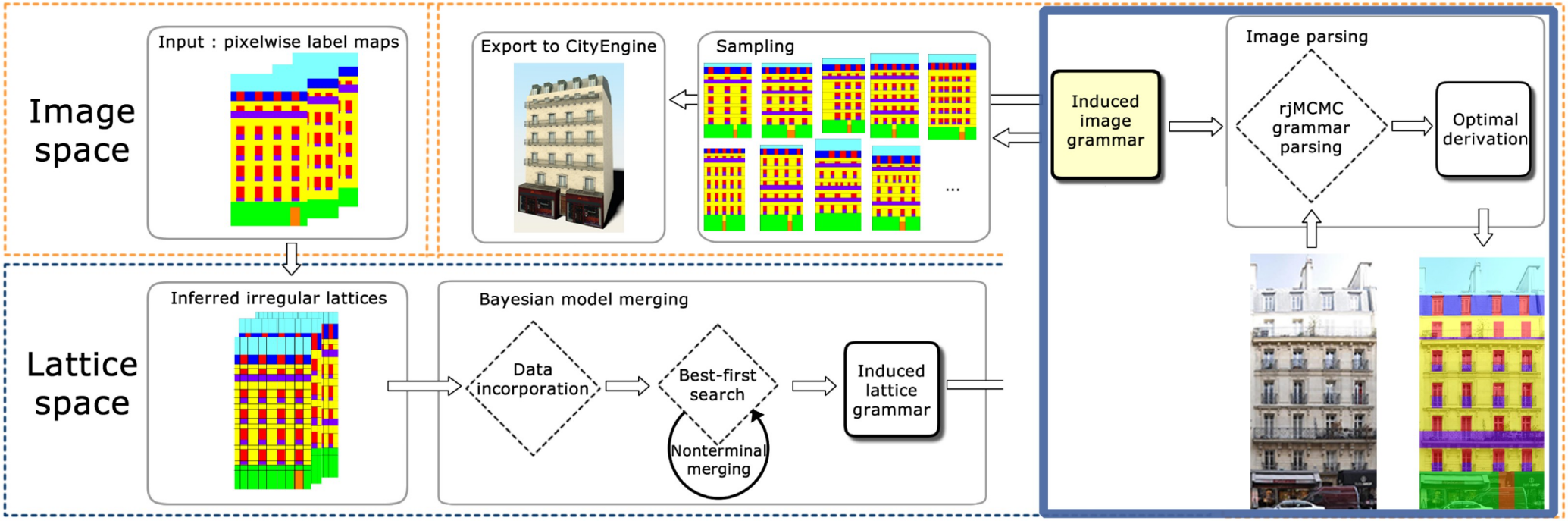
How can we use the learned grammar?

- Parsing existing facades



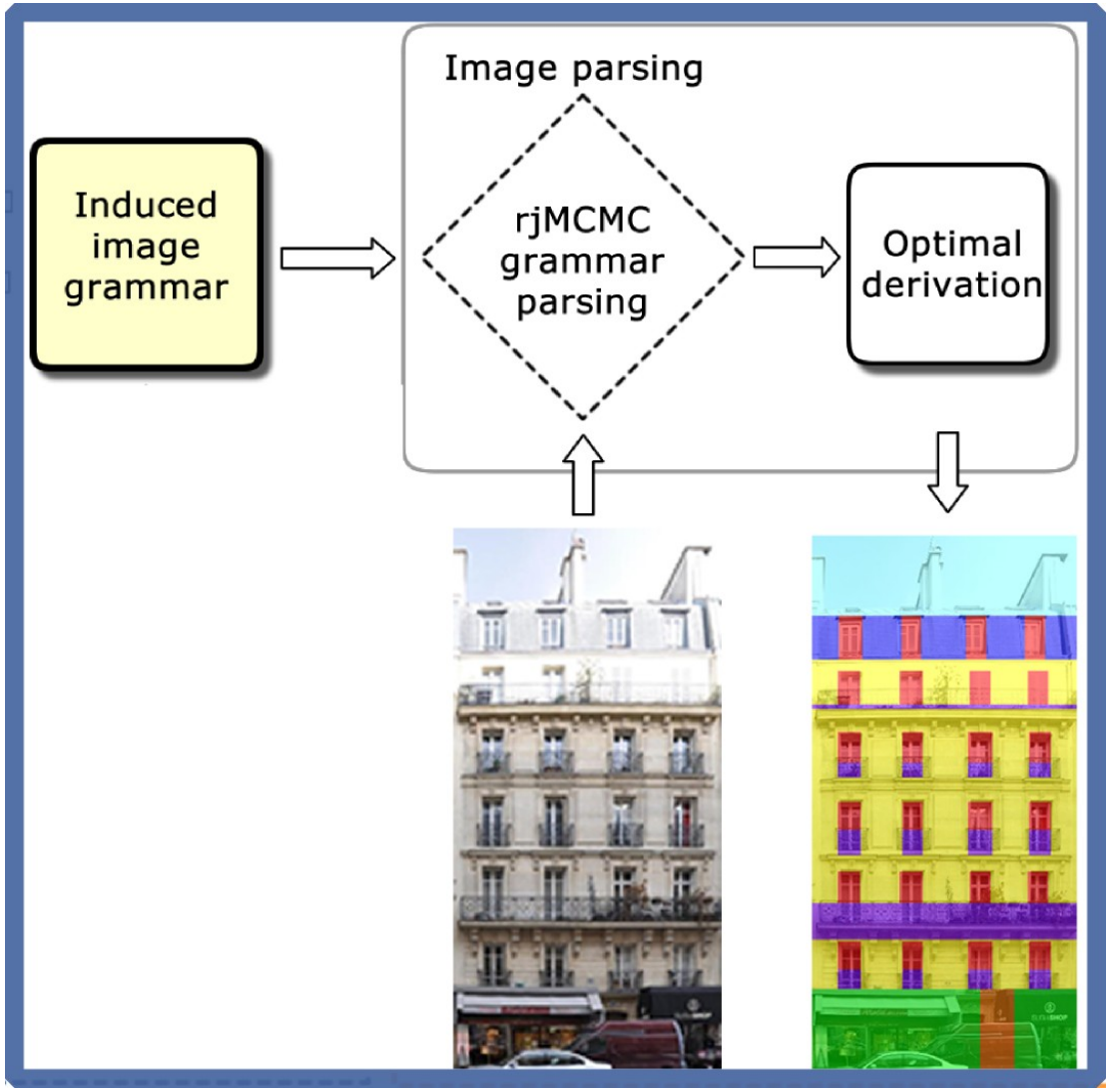
How can we use the learned grammar?

- Parsing existing facades



How can we use the learned grammar?

- Parsing existing facades



What kind of a grammar can we learn?

2D-ASCFG

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

What kind of a grammar can we learn?

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

What kind of a grammar can we learn?

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

What kind of a grammar can we learn?

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

What kind of a grammar can we learn?

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

What kind of a grammar can we learn?

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

- N: Nonterminal symbols
- T: Terminal symbols
- S: Axiom
- R: Production rules

What kind of a grammar can we learn?

Context-Free

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

What kind of a grammar can we learn?

Context-Free

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n$$

What kind of a grammar can we learn?

Stochastic Context-Free

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n$$

What kind of a grammar can we learn?

Stochastic Context-Free

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

Rule probabilities

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n$$

What kind of a grammar can we learn?

Stochastic Context-Free

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

- N: Nonterminal symbols
- T: Terminal symbols
- S: Axiom
- R: Production rules
- P: Rule probabilities

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n \quad [p]$$

KU LEUVEN ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich VISICS CVL Computer Vision Lab

What kind of a grammar can we learn?

Attributed **S**tochastic **C**ontext-**F**ree

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

Rule probabilities

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n \quad [p]$$

KU LEUVEN ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich VISICS CVL Computer Vision Lab

What kind of a grammar can we learn?

Attributed **S**tochastic **C**ontext-**F**ree

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

- N: Nonterminal symbols
- T: Terminal symbols
- S: Axiom
- R: Production rules
- P: Rule probabilities
- A: Rule attributes

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n \quad [p]$$

What kind of a grammar can we learn?

Attributed **S**tochastic **C**ontext-**F**ree

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

Rule attributes

Rule probabilities

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n \quad [p] \qquad \{ \{s_1 s_2 \ldots s_n\}, \ldots \}$$

What kind of a grammar can we learn?

2-Dimensional **A**ttributed **S**tochastic **C**ontext-**F**ree

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

Rule attributes

Rule probabilities

$$X \rightarrow \lambda_1 \lambda_2 \ldots \lambda_n \quad [p] \qquad \{ \{s_1 s_2 \ldots s_n\}, \ldots \}$$

What kind of a grammar can we learn?

2-Dimensional **A**ttributed **S**tochastic **C**ontext-**F**ree

2D-ASCFG

$$G = (N, T, S, R, P, A)$$

Nonterminal symbols

Terminal symbols

Axiom

Production rules

Rule attributes

Rule probabilities

$$X \xrightarrow{H} \lambda_1\lambda_2...\lambda_n \quad [p] \qquad \{\{s_1 s_2...s_n\},...\}$$

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R

$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

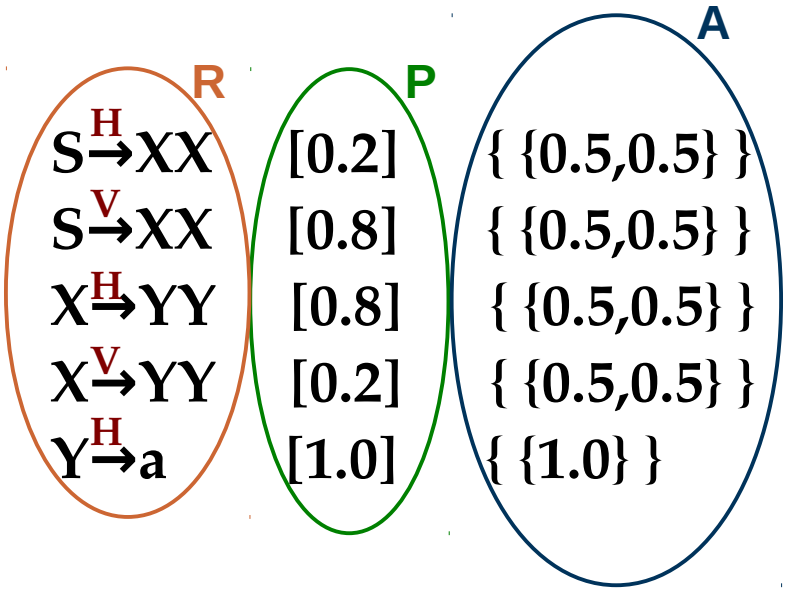
R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}



Input lattice l:

Parsing:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

a	a
a	a

Parsing:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

a	a
a	a

Parsing:

δ_1:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

δ_1:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

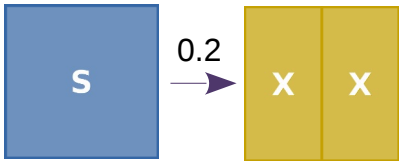
T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

δ_1:



A simple parsing example

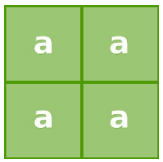
$G = (N, T, S, R, P, A)$

N = {S, X, Y}

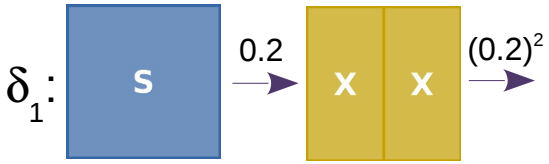
T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:



Parsing:



A simple parsing example

$G = (N, T, S, R, P, A)$

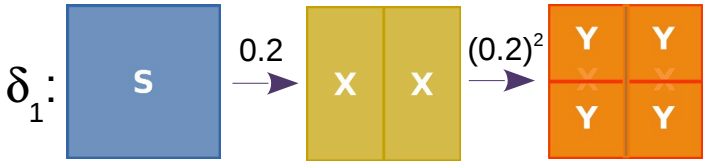
N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



A simple parsing example

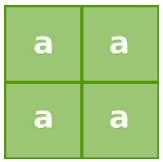
$G = (N, T, S, R, P, A)$

N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:



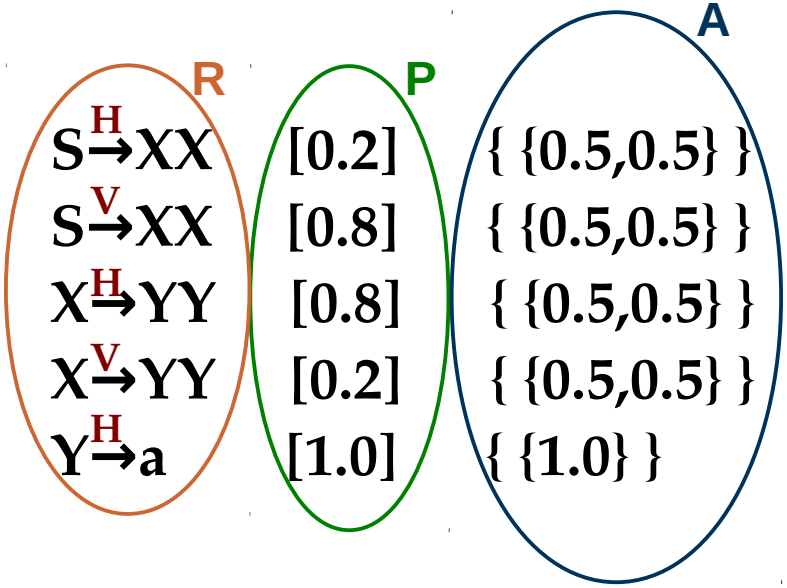
Parsing:

A simple parsing example

$G = (N, T, S, R, P, A)$

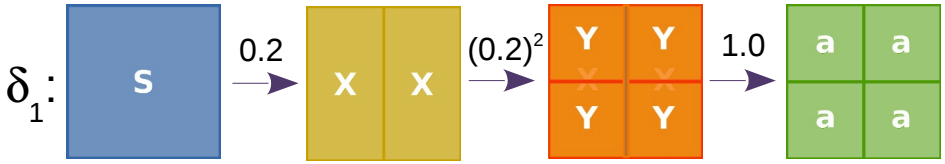
N = {S, X, Y}

T = {a}



Input lattice l:

Parsing:



A simple parsing example

$G = (N, T, S, R, P, A)$

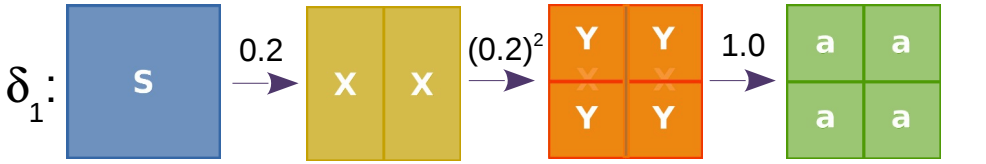
N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$

A simple parsing example

$G = (N, T, S, R, P, A)$

N = {S, X, Y}

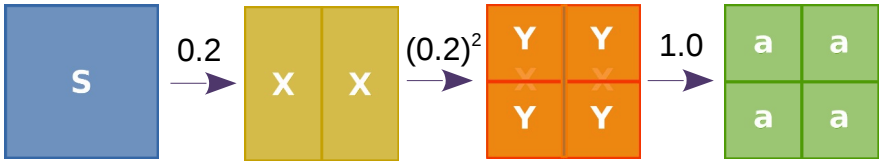
T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

δ_1:



$P(\delta_1)=0.008$

δ_2:

A simple parsing example

$G = (N, T, S, R, P, A)$

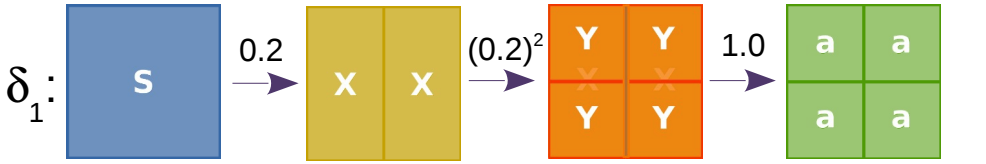
N = {S, X, Y}

T = {a}

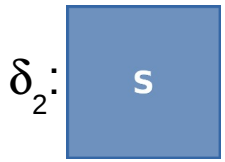
R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$



A simple parsing example

$G = (N, T, S, R, P, A)$

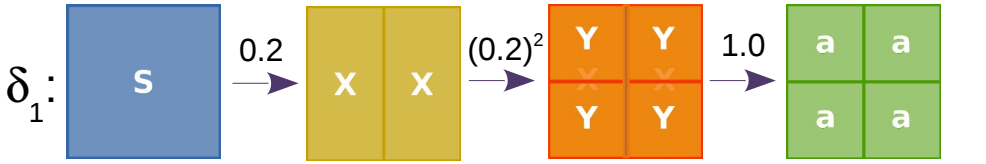
N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$

A simple parsing example

$G = (N, T, S, R, P, A)$

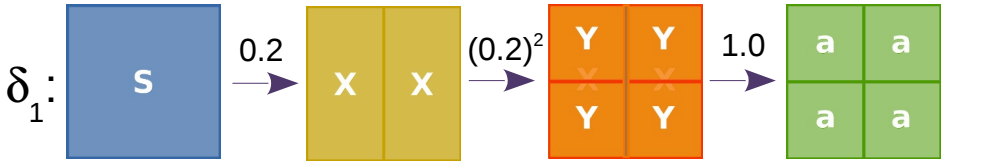
N = {S, X, Y}

T = {a}

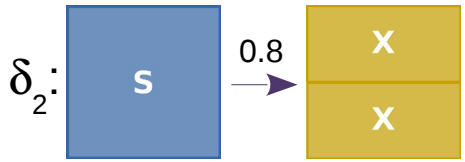
R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$

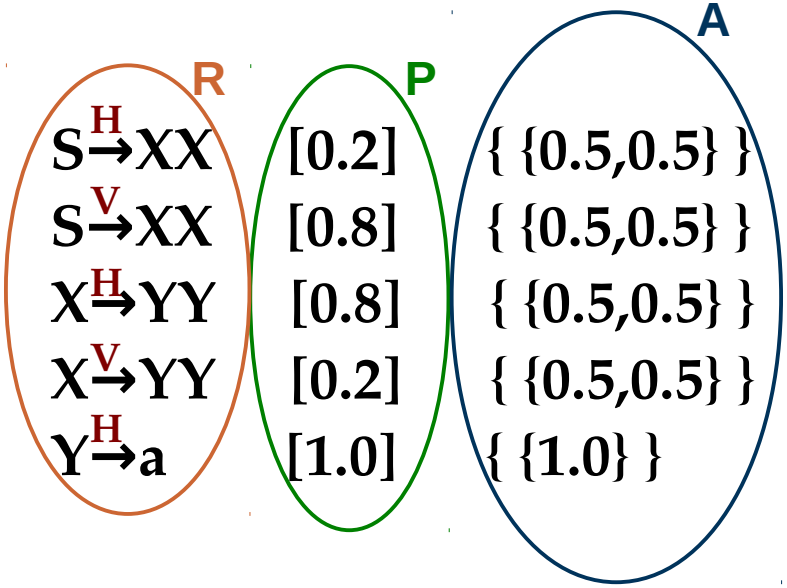


A simple parsing example

$G = (N, T, S, R, P, A)$

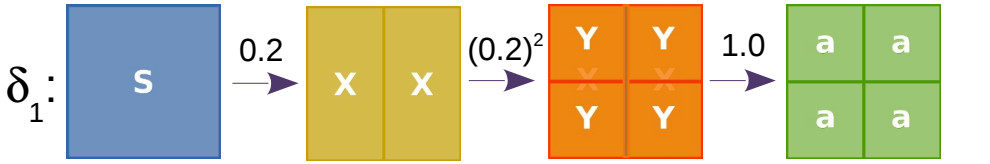
N = {S, X, Y}

T = {a}

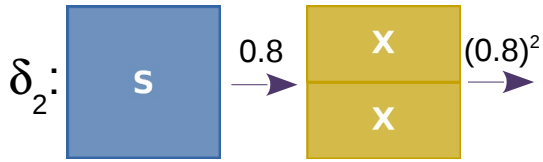


Input lattice l:

Parsing:



$P(\delta_1)=0.008$

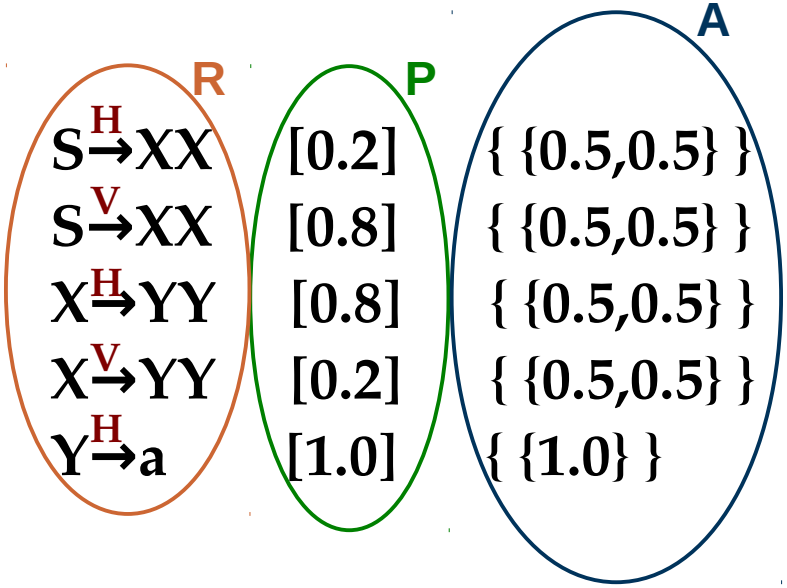


A simple parsing example

$G = (N, T, S, R, P, A)$

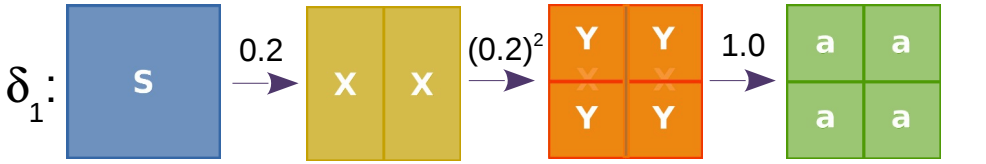
N = {S, X, Y}

T = {a}

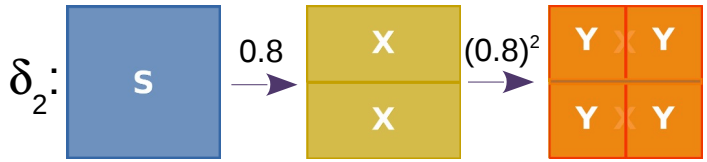


Input lattice l:

Parsing:



$P(\delta_1)=0.008$

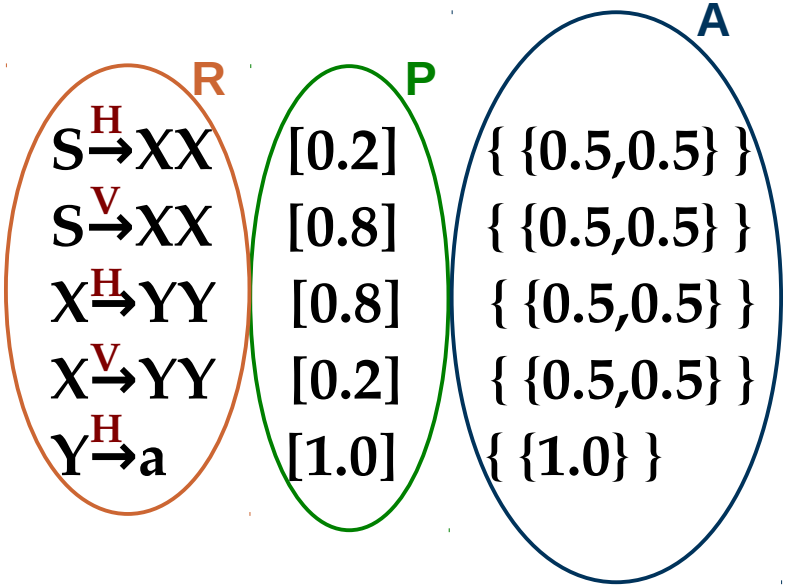


A simple parsing example

$G = (N, T, S, R, P, A)$

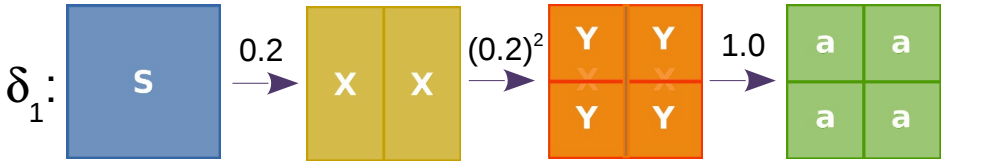
N = {S, X, Y}

T = {a}

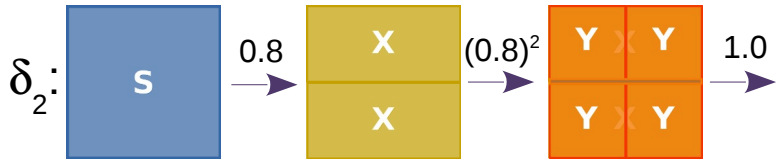


Input lattice l:

Parsing:



$P(\delta_1)=0.008$



A simple parsing example

$G = (N, T, S, R, P, A)$

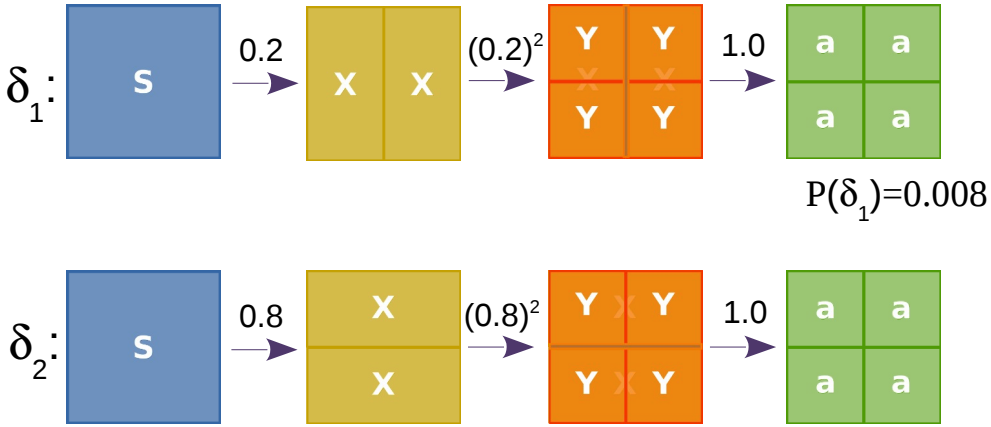
N = {S, X, Y}

T = {a}

R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:

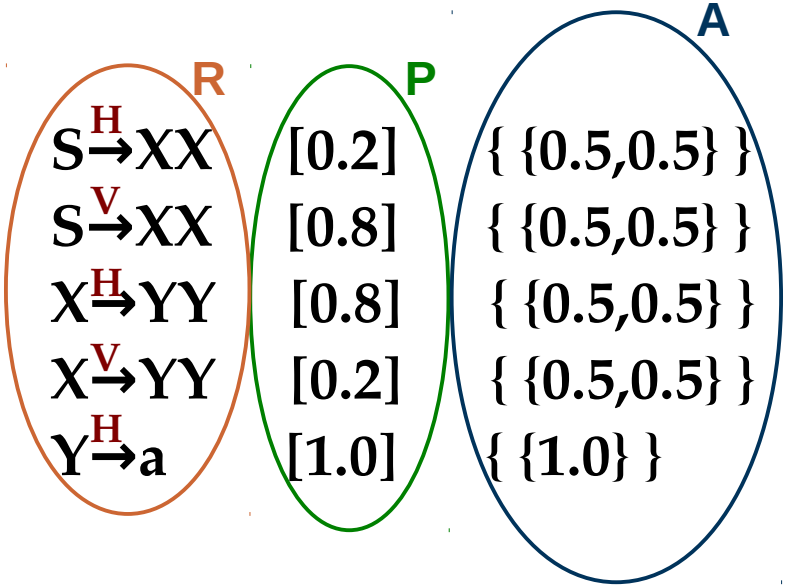


A simple parsing example

$G = (N, T, S, R, P, A)$

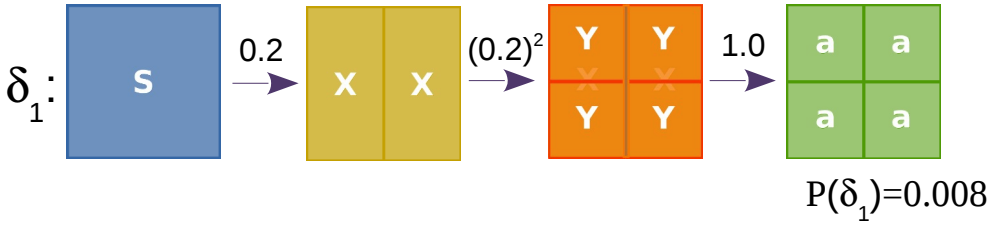
N = {S, X, Y}

T = {a}

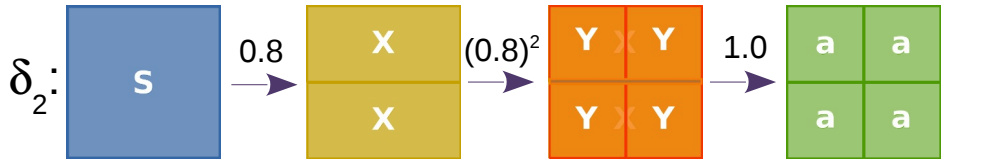


Input lattice l:

Parsing:



$P(\delta_1)=0.008$



$P(\delta_2)=0.512$

A simple parsing example

$G = (N, T, S, R, P, A)$

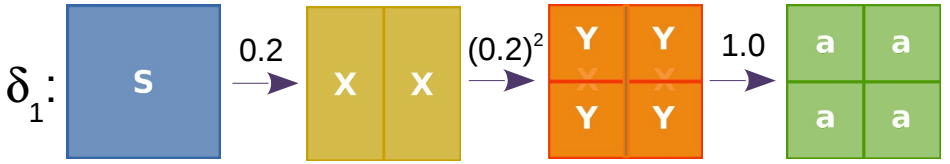
N = {S, X, Y}

T = {a}

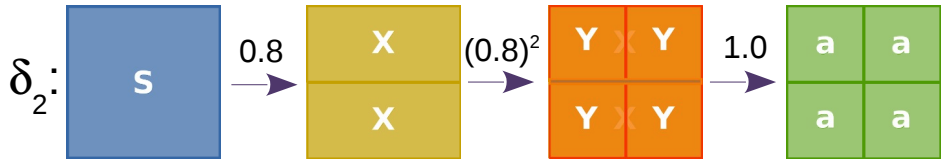
R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$



$P(\delta_2)=0.512$

Input likelihood: $L(l \mid G) = \sum_{\delta \Rightarrow l} P(\delta) = 0.52$

A simple parsing example

$G = (N, T, S, R, P, A)$

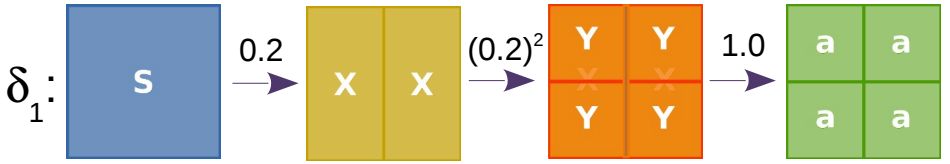
N = {S, X, Y}

T = {a}

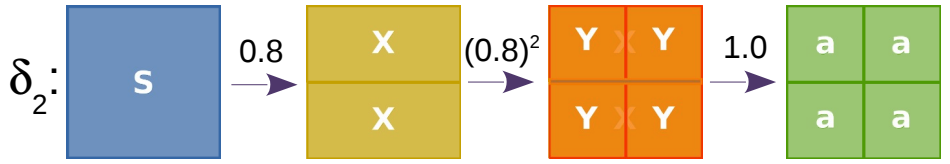
R	P	A
$S \xrightarrow{H} XX$	[0.2]	{ {0.5,0.5} }
$S \xrightarrow{V} XX$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{H} YY$	[0.8]	{ {0.5,0.5} }
$X \xrightarrow{V} YY$	[0.2]	{ {0.5,0.5} }
$Y \xrightarrow{H} a$	[1.0]	{ {1.0} }

Input lattice l:

Parsing:



$P(\delta_1)=0.008$

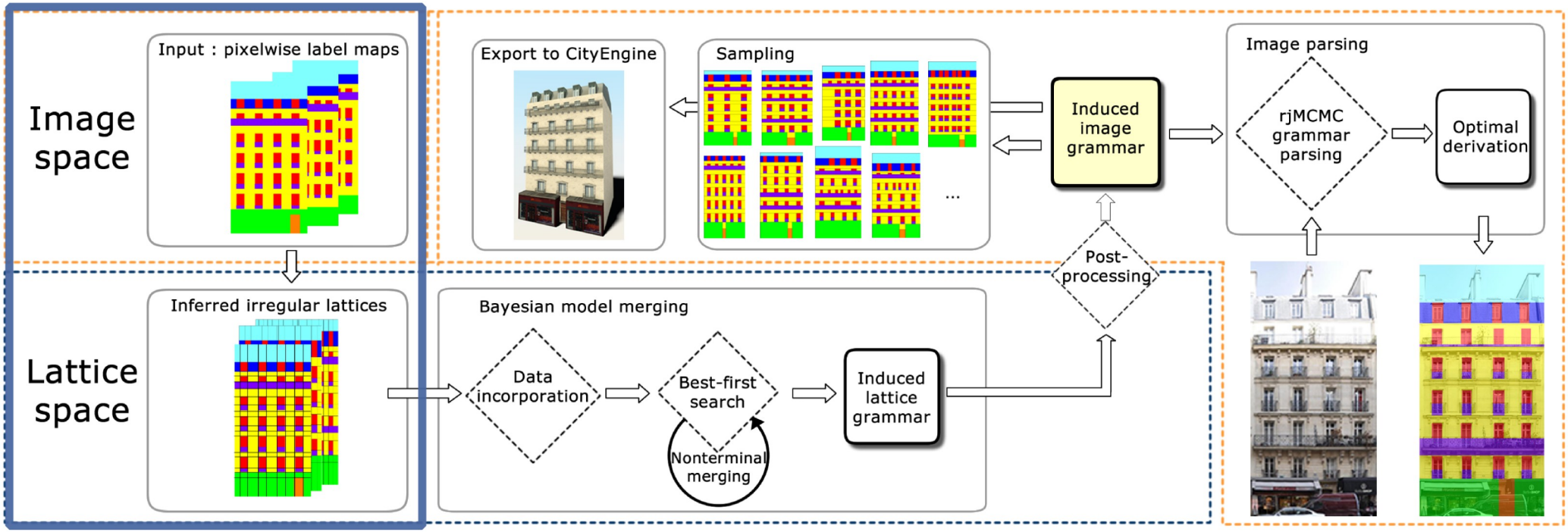


$P(\delta_2)=0.512$

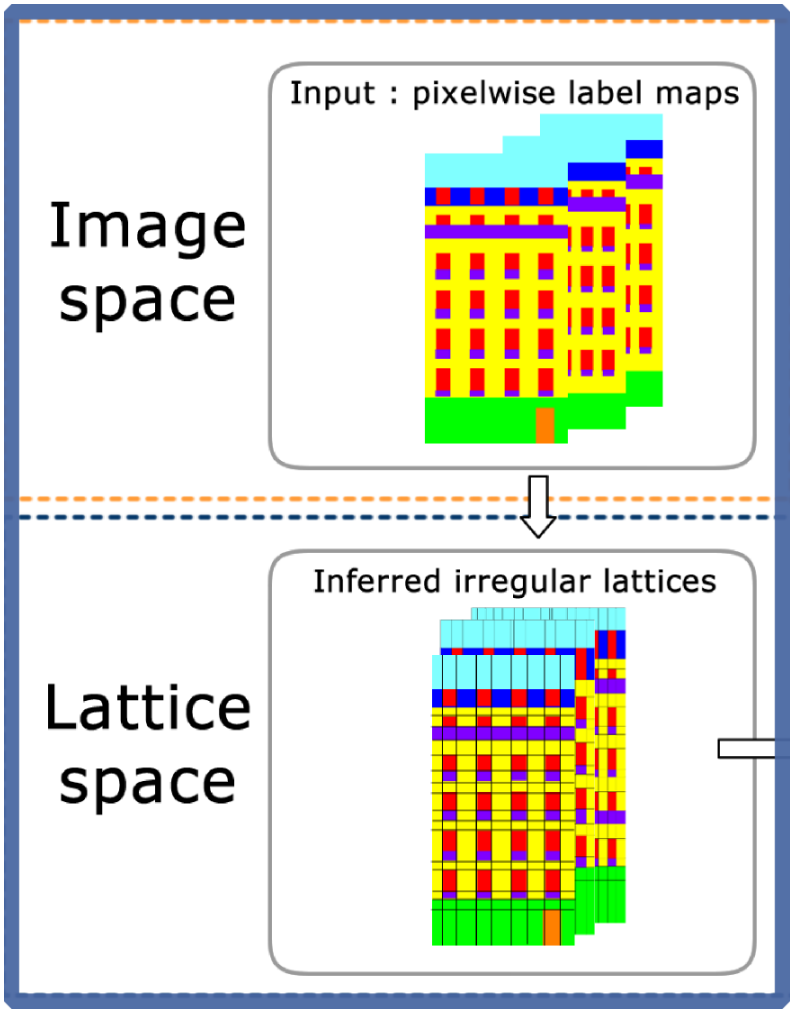
Input likelihood: $L(l \mid G) = \sum_{\delta \Rightarrow l} P(\delta) = 0.52$

Viterbi approximation: $L(l \mid G) = \max_{\delta \Rightarrow l} P(\delta) = 0.512$

Method overview

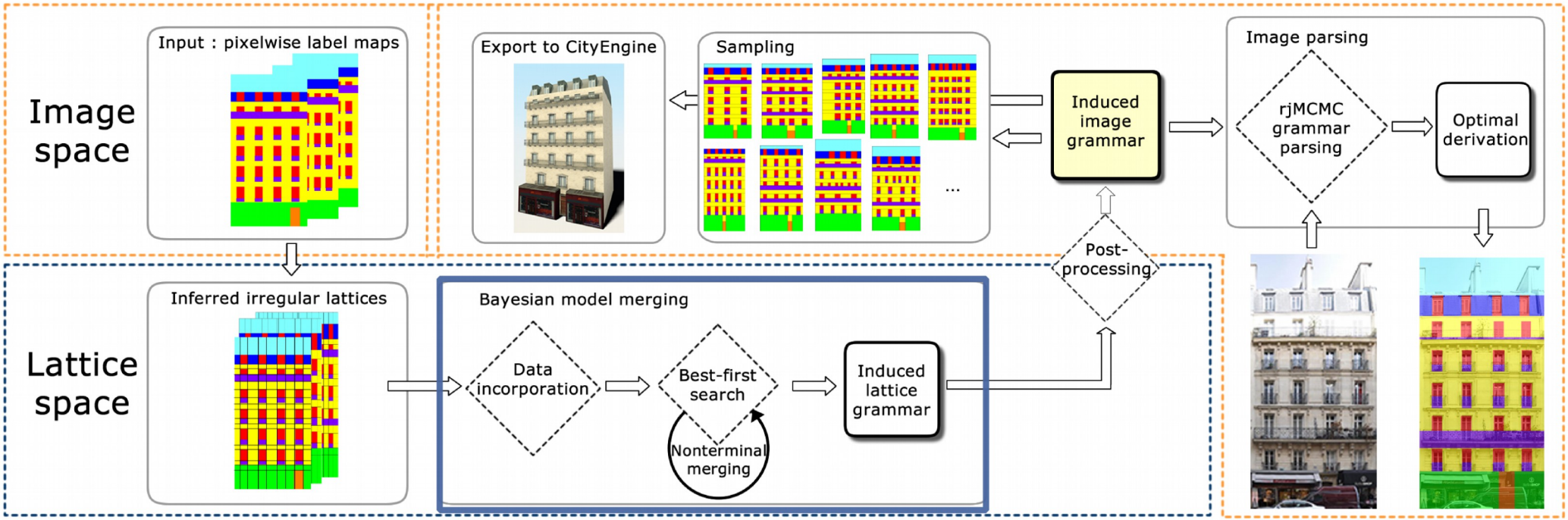


Lattice creation



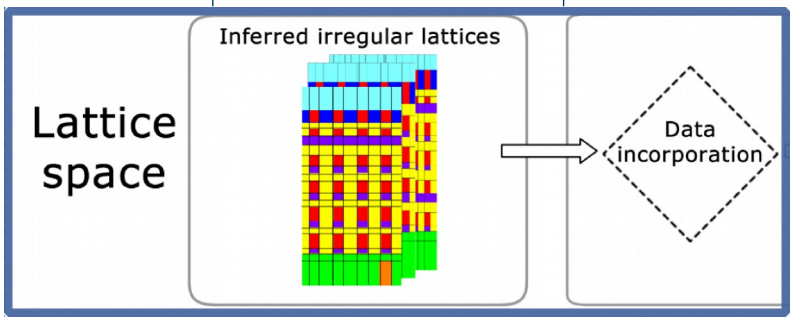
- Approach similar to Riemenschneider et al. (2012)
- Every rectangular region (tile) is labeled with the majority vote from the corresponding pixel labels

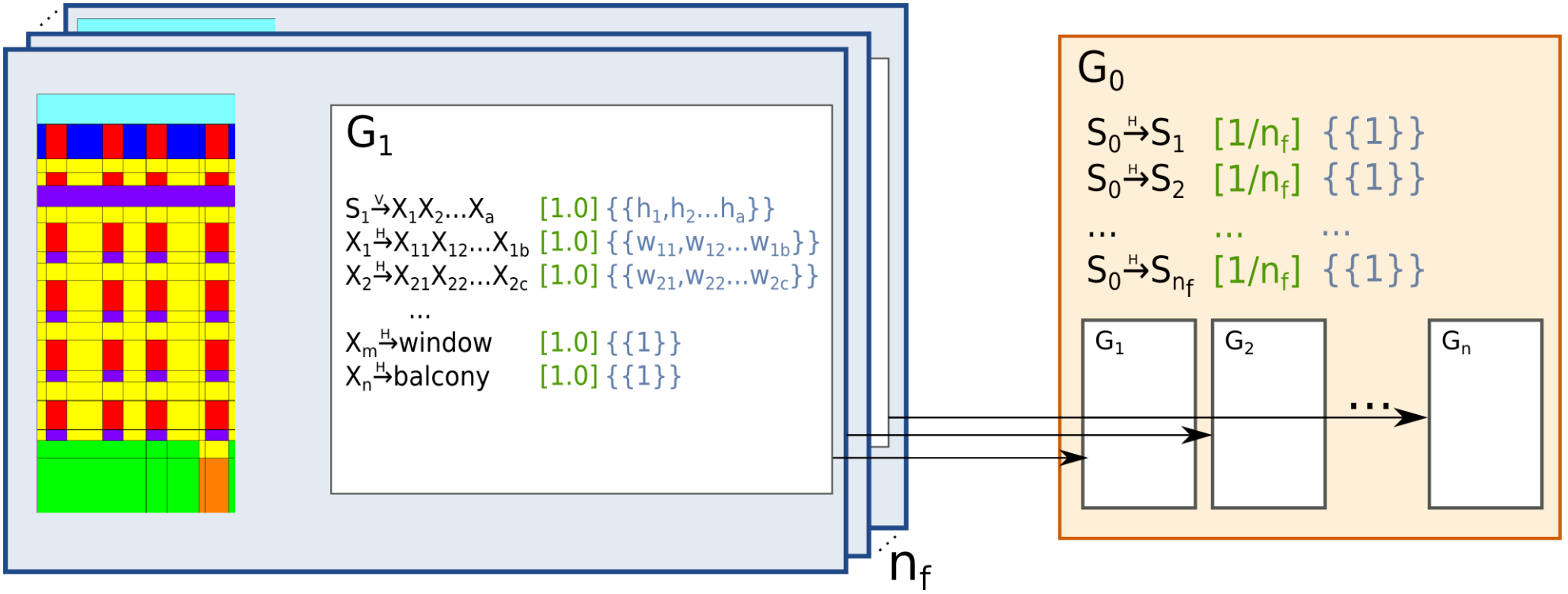
Bayesian model merging



- **Data incorporation**: given a body of data, build an initial grammar which generates only the input examples.
- **Model merging**: propose a candidate grammar by altering the structure of the currently best grammar.
- **Model evaluation**: evaluate the fitness of the candidate grammar compared to the currently best grammar.
- **Search**: use model merging to explore the grammar space, searching for the optimal grammar

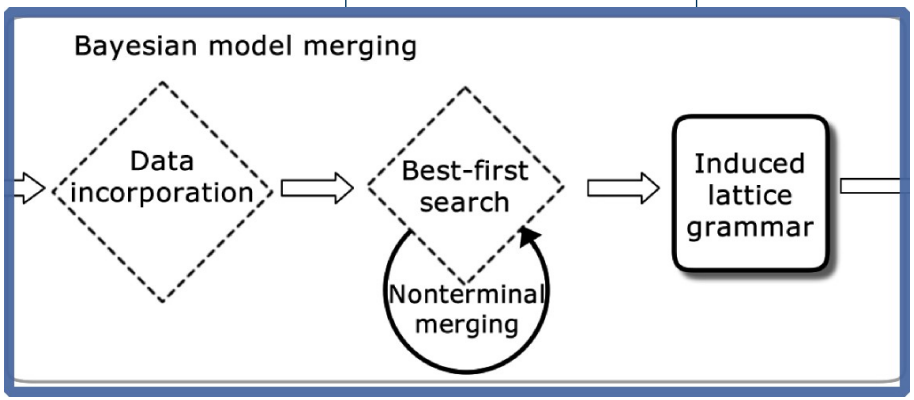
Data incorporation





KU LEUVEN ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Nonterminal merging: Proposing candidate grammars



- Two nonterminals X_1 and X_2 are selected from the current grammar and replaced with a new nonterminal Y
  - RHS occurrences are replaced:

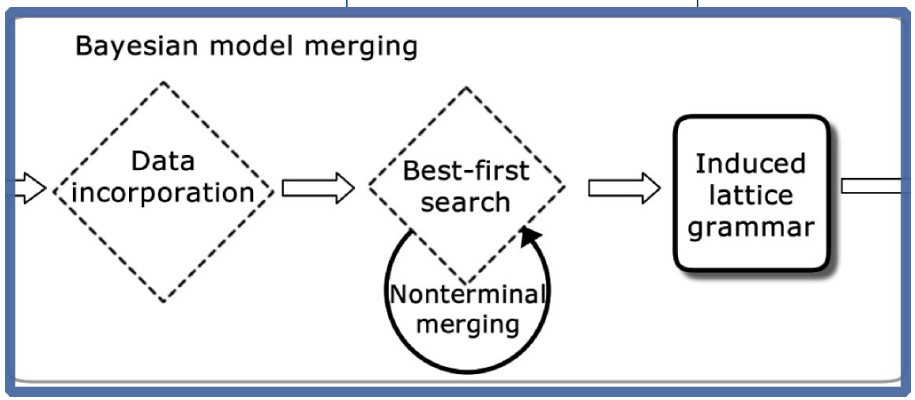
$$\begin{matrix} Z_1 \rightarrow \mu_1 X_1 \lambda_1 \\ Z_2 \rightarrow \mu_2 X_2 \lambda_2 \end{matrix} \xrightarrow{\text{merge}} \begin{matrix} Z_1 \rightarrow \mu_1 Y \lambda_1 \\ Z_2 \rightarrow \mu_2 Y \lambda_2 \end{matrix}$$

  - If $Z_1=Z_2$, $\mu_1=\mu_2$, $\lambda_1=\lambda_2$, productions are merged and their attribute sets joined
  - LHS occurrences are also replaced:

$$\begin{matrix} X_1 \rightarrow \lambda_1 \\ X_2 \rightarrow \lambda_2 \end{matrix} \xrightarrow{\text{merge}} \begin{matrix} Y \rightarrow \lambda_1 \\ Y \rightarrow \lambda_2 \end{matrix}$$

  - Productions of the form $Y \rightarrow Y$ are removed
  - Restriction: X_1 and X_2 must be 'label-compatible'

Evaluating candidate grammars

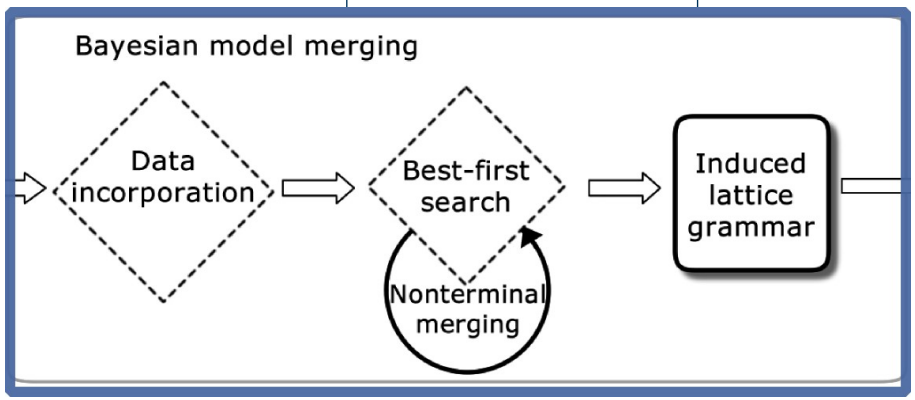


- Goal: find the best trade-off between fit to the input data D and a general preference for simpler models (MDL)

- From a Bayesian perspective:

$$\textit{maximize}\ \ P(G \mid D) \sim P(G) \cdot P(D \mid G)$$

posterior prior likelihood



Evaluating candidate grammars

Prior term

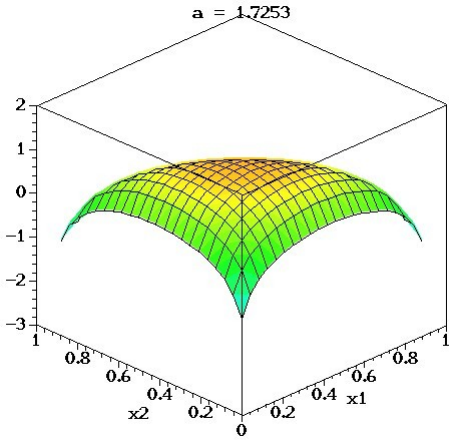
$$P(G)=P(G_s)\cdot P(\theta_g \mid G_s)$$

Structure prior Parameter prior

- Structure prior follows the Minimum Description Length (MDL) principle
  - $P(G_s) = e^{-DL(G_s)}$
  - Grammar description length (in bits):

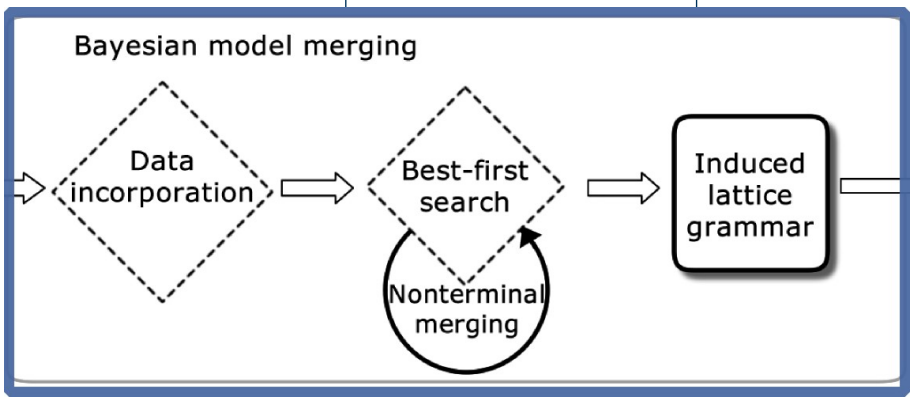
$$DL(G_s)=\Sigma_{r\in R}\ \Sigma_{n\in(r\cap N)}\log|N|$$

- Parameter prior is a symmetrical Dirichlet

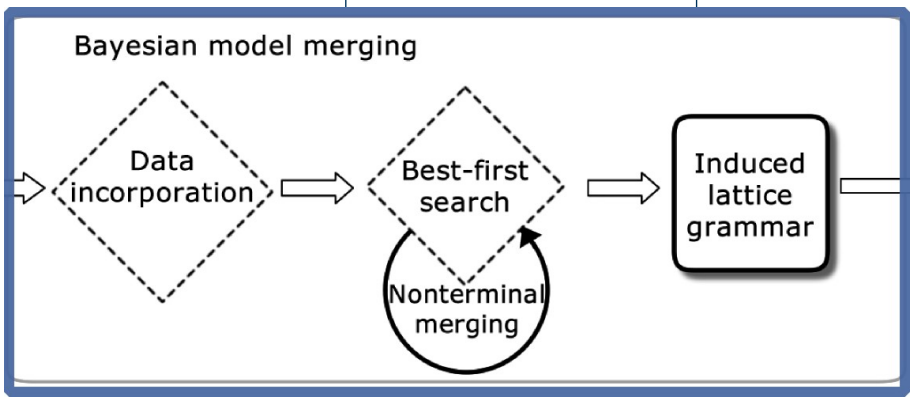


Evaluating candidate grammars

Likelihood term



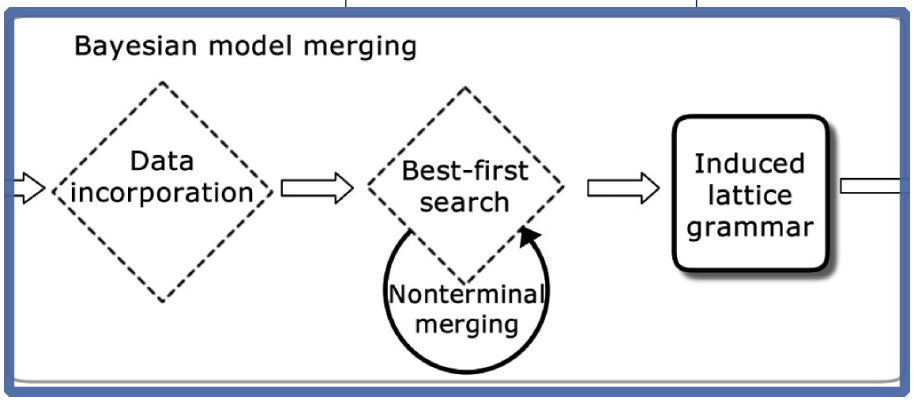
Evaluating candidate grammars



Likelihood term

- To calculate the likelihood, one must integrate over the parameter prior...

- ...or, use the Viterbi assumption:
  - Every input sample is generated by a **single** derivation tree of the grammar
  - The likelihood of that sample is then the product of all rule probabilities used in the Viterbi derivation
  - However, both the Viterbi derivations and the optimal values of rule probabilities θ_g are unknown!



Evaluating candidate grammars

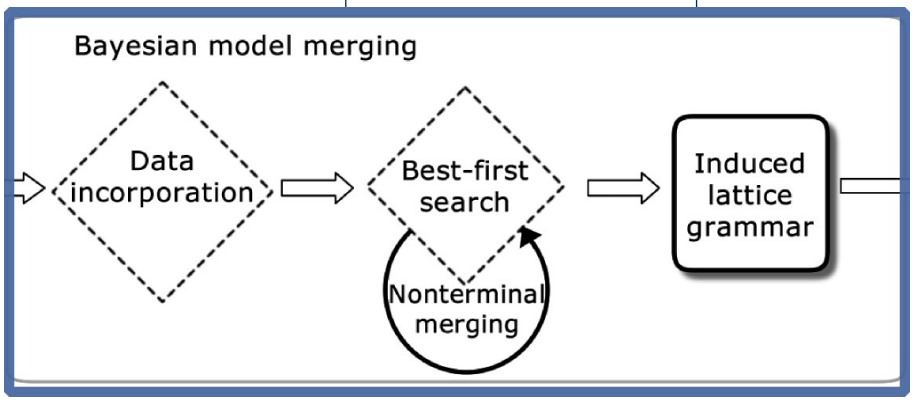
Likelihood term

- To calculate the likelihood, one must integrate over the parameter prior...

$$P(D|G_s) = \int_{\theta_g} P(\theta_g|G_s)P(D|G_s,\theta_g)d\theta_g$$

- ...or, use the Viterbi assumption:
  - Every input sample is generated by a **single** derivation tree of the grammar
  - The likelihood of that sample is then the product of all rule probabilities used in the Viterbi derivation
  - However, both the Viterbi derivations and the optimal values of rule probabilities θ_g are unknown!

Evaluating candidate grammars

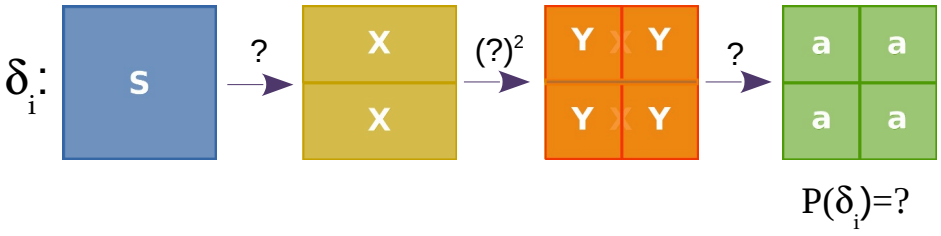


Likelihood term

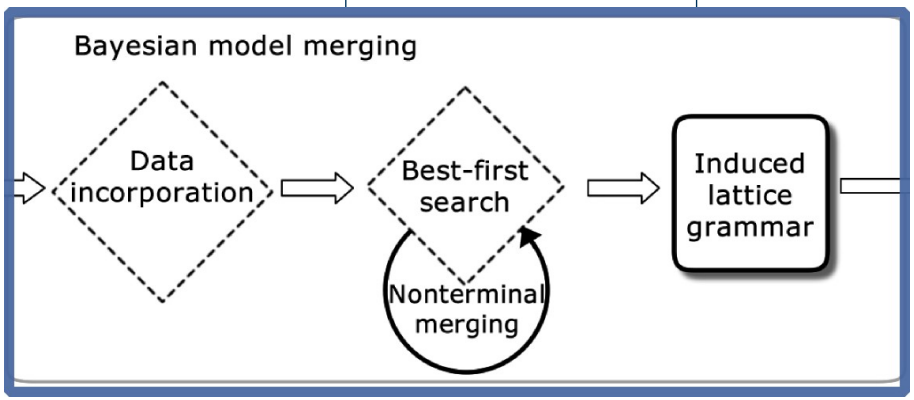
- To calculate the likelihood, one must integrate over the parameter prior...

$$P(D|G_s) = \int_{\theta_g} P(\theta_g|G_s)P(D|G_s,\theta_g)d\theta_g$$

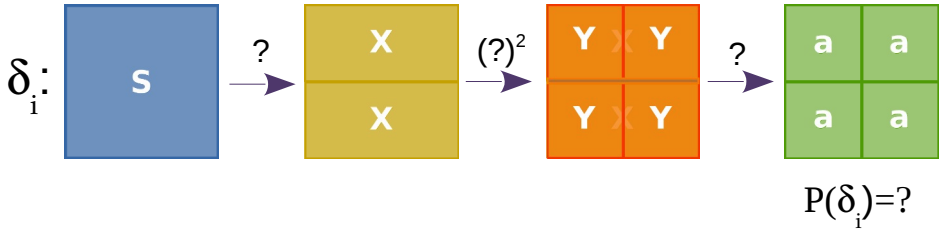
- ...or, use the Viterbi assumption:
  - Every input sample is generated by a **single** derivation tree of the grammar
  - The likelihood of that sample is then the product of all rule probabilities used in the Viterbi derivation
  - However, both the Viterbi derivations and the optimal values of rule probabilities θ_g are unknown!



Evaluating candidate grammars

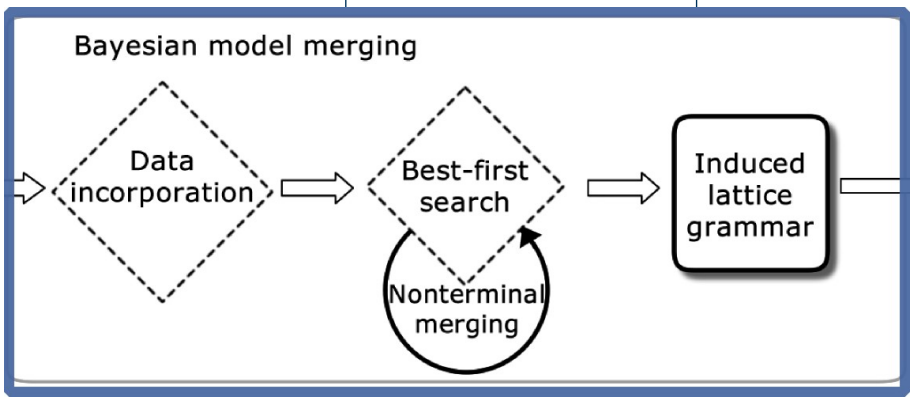


Likelihood term

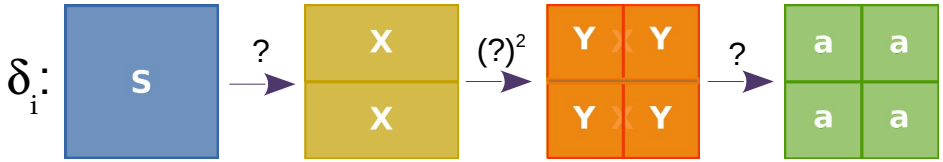


$P(\delta_i)=?$

Evaluating candidate grammars

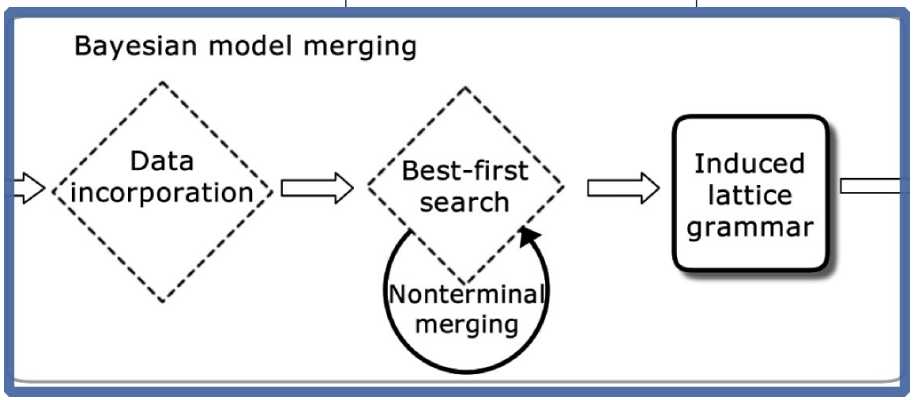


Likelihood term

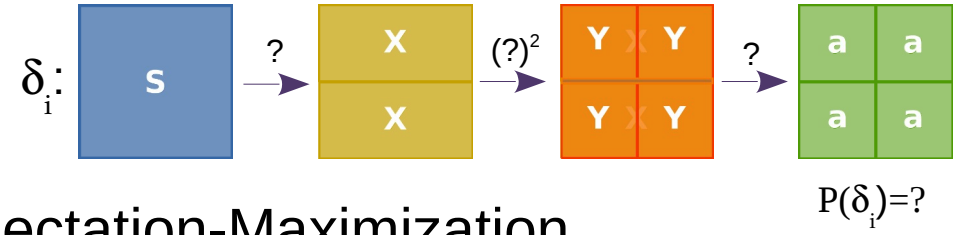


- Solution: Expectation-Maximization
- **E-step**: starting from an estimate of θ_g, calculate the expected usage counts of each rule
  - Find Viterbi derivations of all input data, count the number of times each rule is used
- **M-step: r**e-estimate the rule probabilities θ_g:

Evaluating candidate grammars

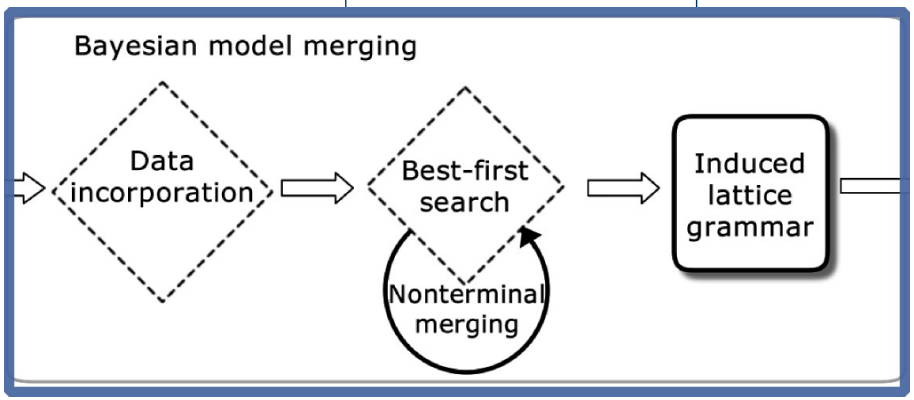


Likelihood term

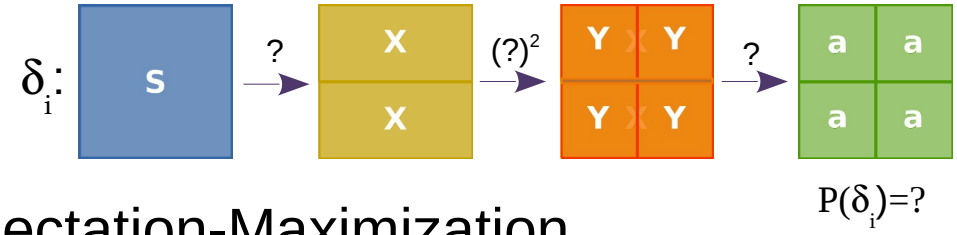


- Solution: Expectation-Maximization
- **E-step**: starting from an estimate of θ_g, calculate the expected usage counts $\hat{c}(X \rightarrow \lambda)$ of each rule
  - Find Viterbi derivations of all input data, count the number of times each rule is used
- **M-step: r**e-estimate the rule probabilities θ_g:

Evaluating candidate grammars



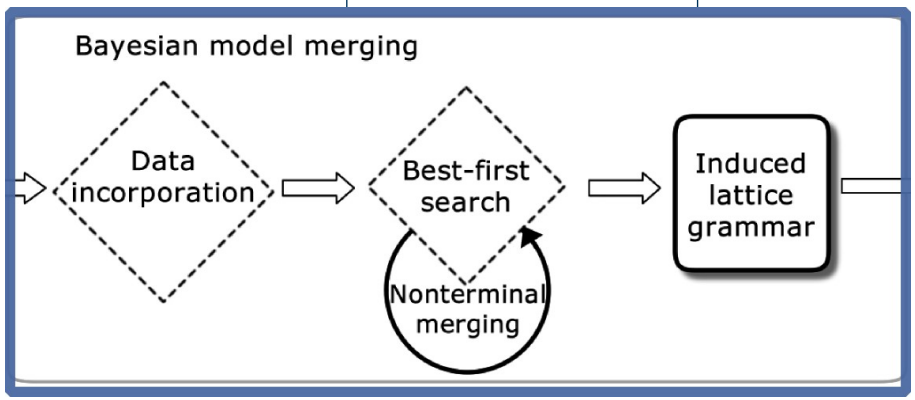
Likelihood term



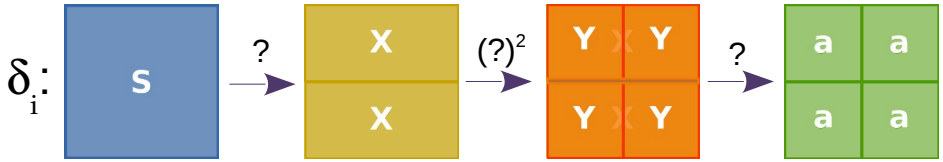
- Solution: Expectation-Maximization
- **E-step**: starting from an estimate of θ_g, calculate the expected usage counts $\hat{c}(X \rightarrow \lambda)$ of each rule
  - Find Viterbi derivations of all input data, count the number of times each rule is used
- **M-step: r**e-estimate the rule probabilities θ_g:

KU LEUVEN ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Evaluating candidate grammars



Likelihood term



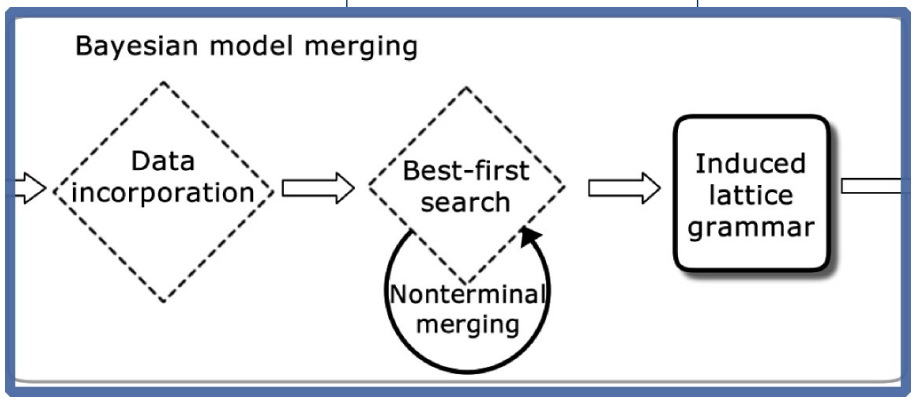
- Solution: Expectation-Maximization
- **E-step**: starting from an estimate of θ_g, calculate the expected usage counts $\hat{c}(X \to \lambda)$ of each rule
  - Find Viterbi derivations of all input data, count the number of times each rule is used
- **M-step: r**e-estimate the rule probabilities θ_g:

$$\hat{P}(X \to \lambda) = \frac{\hat{c}(X \to \lambda)}{\sum_{\mu} \hat{c}(X \to \mu)}$$

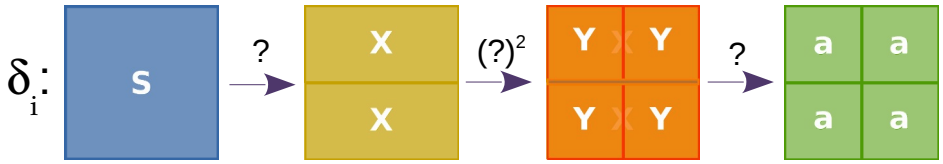
KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Evaluating candidate grammars

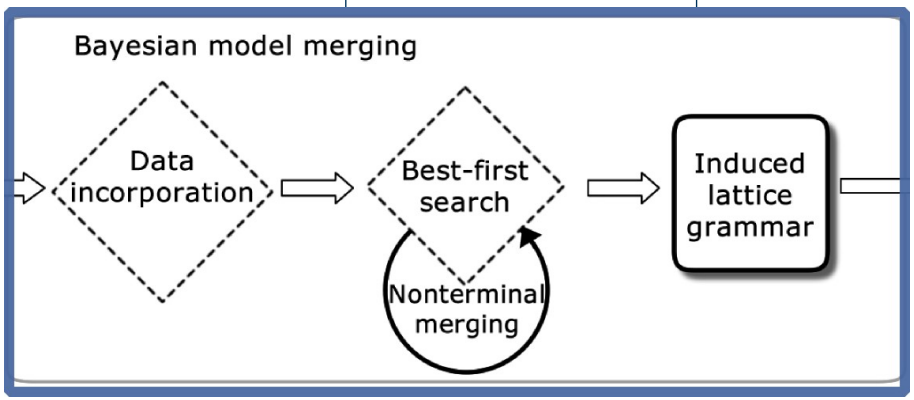


2D Earley parsing

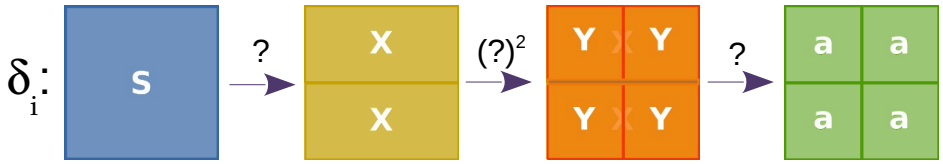


$P(\delta_i)=?$

Evaluating candidate grammars



2D Earley parsing

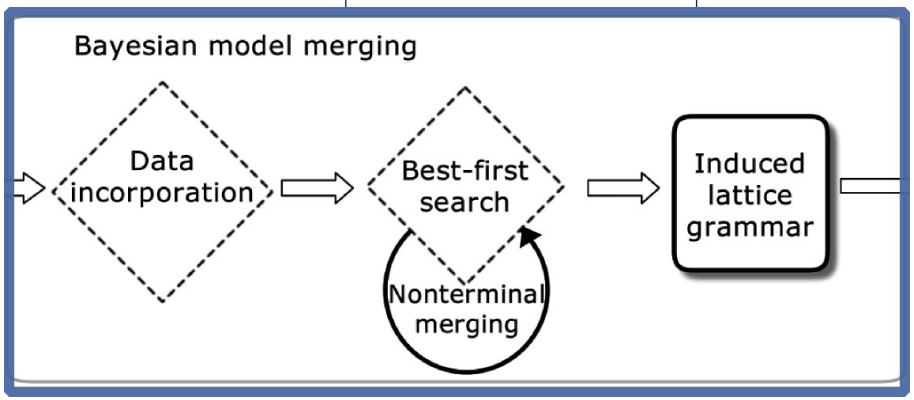


$P(\delta_i)=?$

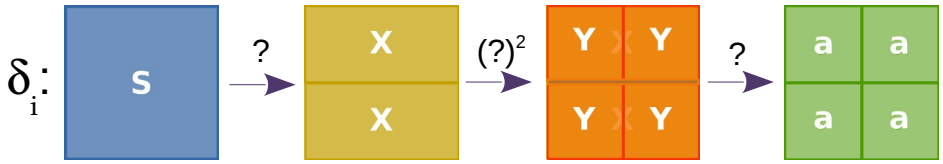
- How to find the Viterbi derivations in the E-step?
- Generalizing the Earley's string parser (Earley, 1970) to 2D
  - Top-down parsing algorithm
  - Worst-case complexity is $O(n^3)$, but usually performs better
    - e.g. $O(n^2)$ for unambiguous grammars
  - Does not require the grammar to be in CNF
  - Calculation of Viterbi probabilities with the extension of (Stolcke, 1994)

KU LEUVEN ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Evaluating candidate grammars



2D Earley parsing

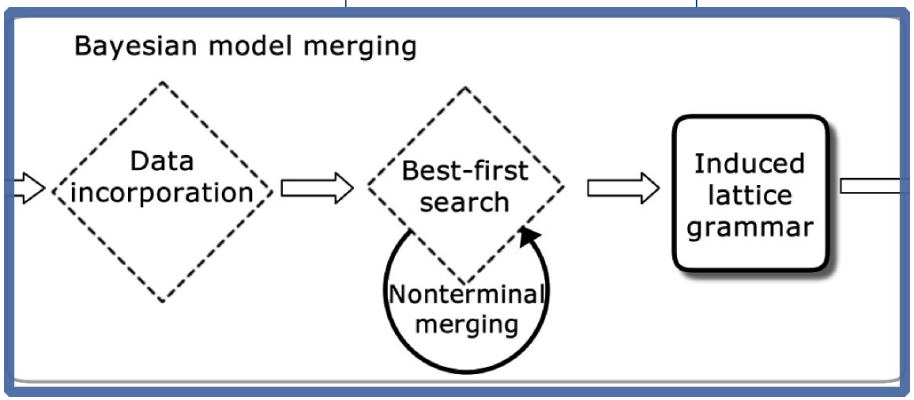


$P(\delta_i)=?$

- How to find the Viterbi derivations in the E-step?
- Generalizing the Earley's string parser (Earley, 1970) to 2D
  - Top-down parsing algorithm
  - Worst-case complexity is $O(n^3)$, but usually performs better
    - e.g. $O(n^2)$ for unambiguous grammars
  - Does not require the grammar to be in CNF
  - Calculation of Viterbi probabilities with the extension of (Stolcke, 1994)

A. Martinović and L. Van Gool. Earley parsing for 2D stochastic context free grammars. Technical Report KUL/ESAT/PSI/1301, KU Leuven, 2013.

Search in model space



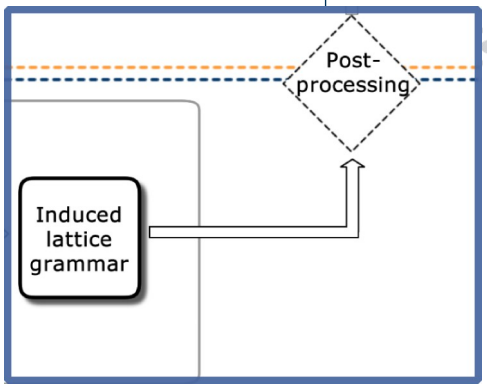
- Posterior calculation is modified using a global prior weight *w*
- Minimizing the energy

$$E(G \mid D) = - w \log P(G) - \log P(D \mid G)$$

  - *w* small: low generalization, search procedure stops earlier
  - *w* large: increasing the tendency to generalize beyond data

- Greedy best-first search
  - In each iteration, every pair of nonterminals is considered for merging
  - All candidate grammars are evaluated
  - The candidate grammar with minimum energy is accepted if this energy is lower than the current grammar's
  - The rule probabilities are learned in each step using the EM procedure

Obtaining the final model



- Casting the induced grammar back to image space
  1. Sequences of the same nonterminal symbol are collapsed, e.g.

$X \rightarrow \lambda YYy \;\; \{\{s_1, y_1, y_2, s_2\}\} \xrightarrow{\text{collapse}} X \rightarrow \lambda Yy \;\; \{\{s_1, y_1 + y_2, s_2\}\}$

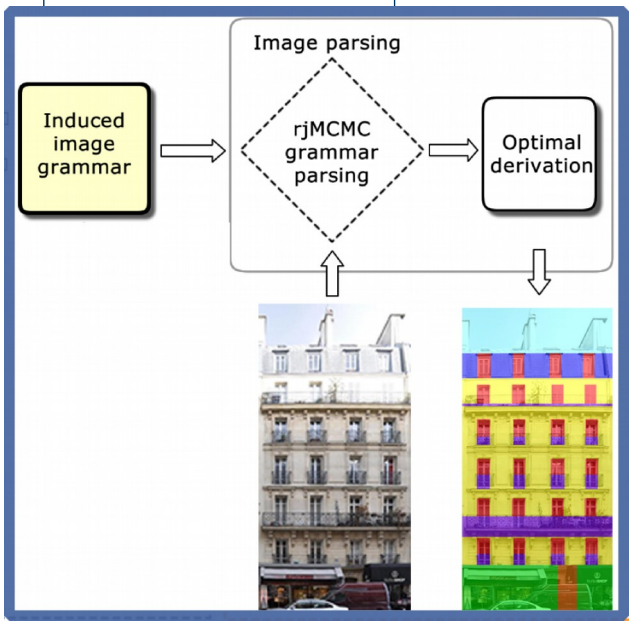
  2. For every rule $p = (X \rightarrow \lambda_1 \ldots \lambda_k)$, we fit a (k-1)-variate Gaussian distribution $\phi(A) = \mathcal{N}(\bar{\mu}, \hat{\Sigma})$ to the set of its attributes $A(p) = \{\alpha_1 \ldots \alpha_n\}$

$$\bar{\mu} = \frac{1}{n} \sum_{j=1}^{n} \alpha_j$$

$$\hat{\Sigma} = \frac{1}{n-1} \sum_{j=1}^{n} (\alpha_j - \bar{\mu})(\alpha_j - \bar{\mu})^T$$

  - This enables us to sample productions with continuous attributes in image space

Parsing in image space



- Find the grammar derivation δ which optimally describes the image
- Optimization is difficult:
  1. Rule attributes can have continuous values
     - Solution: Markov Chain Monte Carlo (MCMC) reduces optimization to sampling
  2. Grammar is stochastic, the number of rules per derivation can change
     - Solution: Reversible jump MCMC (rjMCMC)
- Posterior of a derivation δ given an image I:

$$P(\delta|I) \propto \underbrace{P(I|\delta)}_{\text{Likelihood}} \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Prior

Parsing in image space

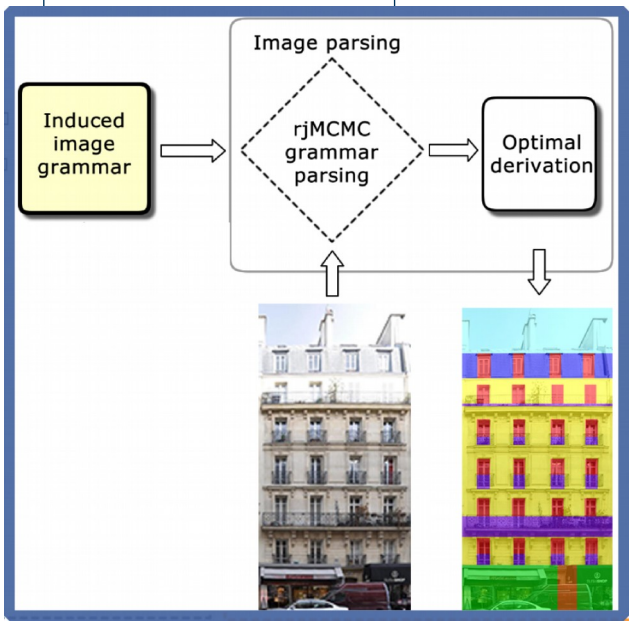
$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term

Attribute term

Likelihood

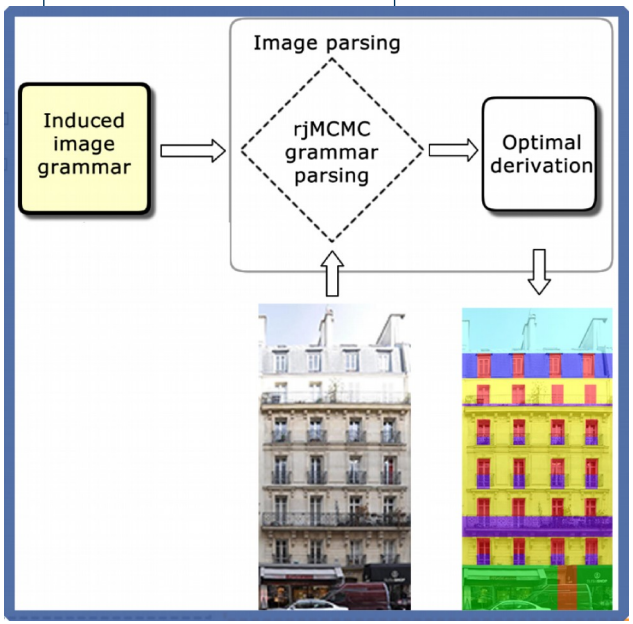
Prior



Parsing in image space

$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

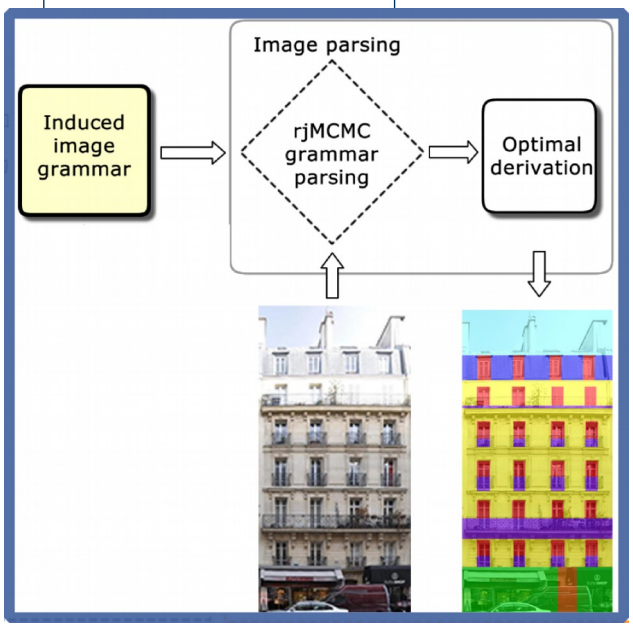
Rule term — Attribute term — Likelihood — Prior



- Boltzmann's transformation:

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

Parsing in image space



Rule term Attribute term

$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Likelihood Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

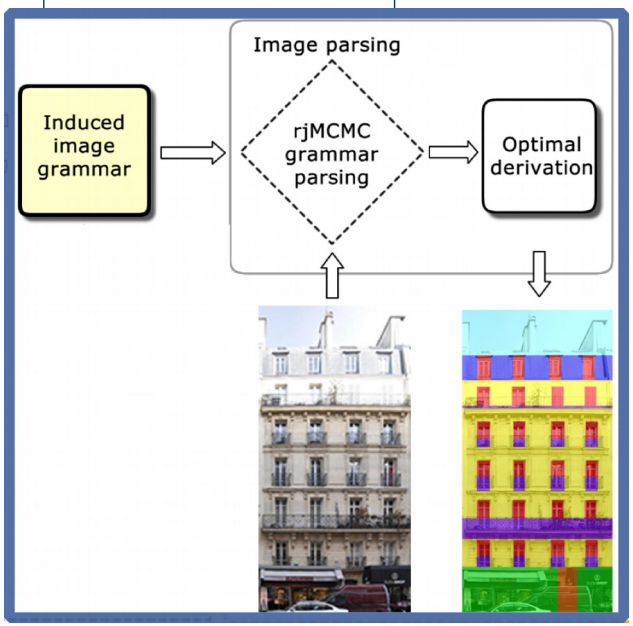
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Likelihood — Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

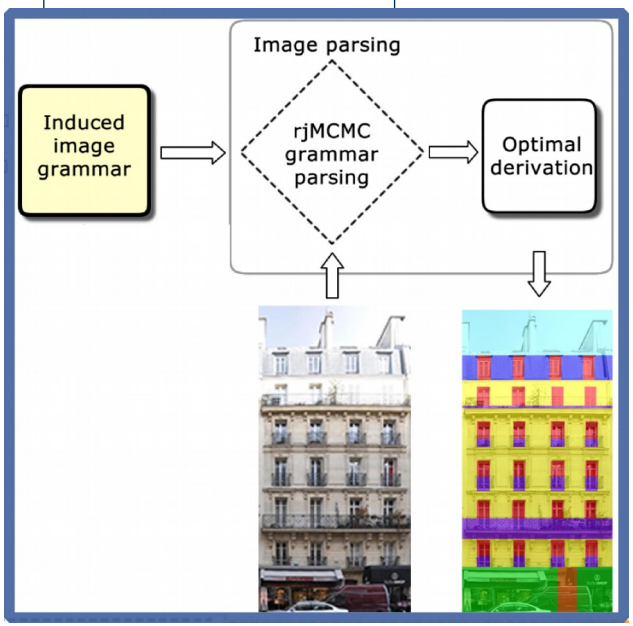
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Likelihood — Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

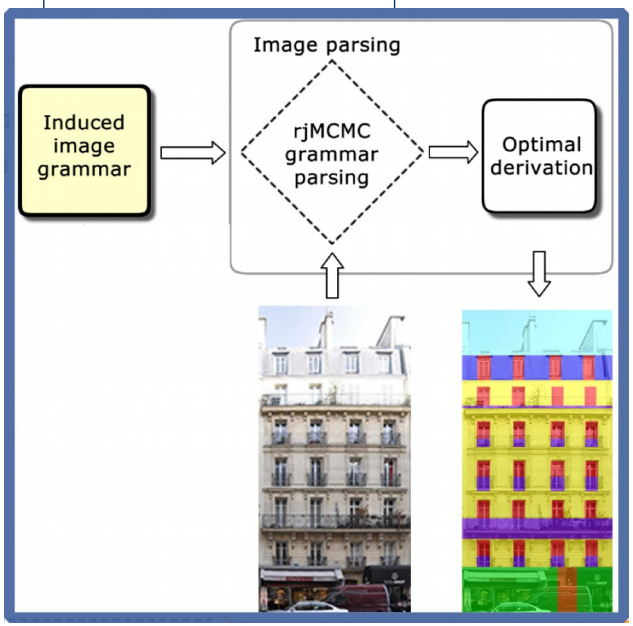
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Likelihood — Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

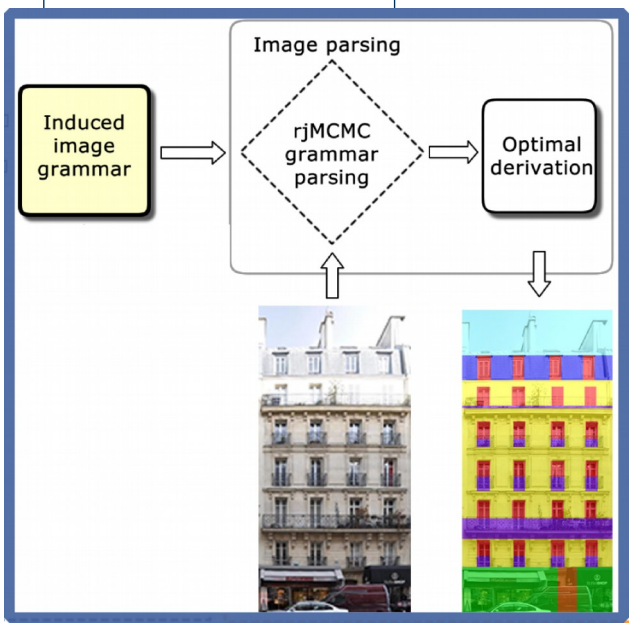
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



Rule term Attribute term

$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Likelihood Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

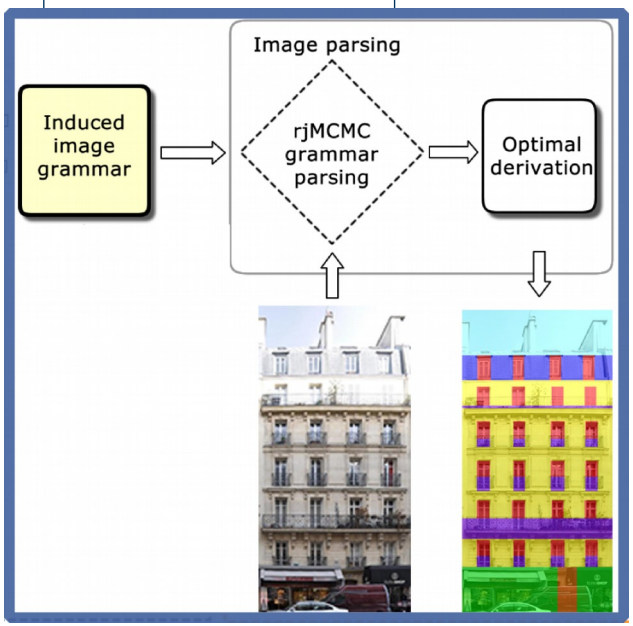
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Likelihood — Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

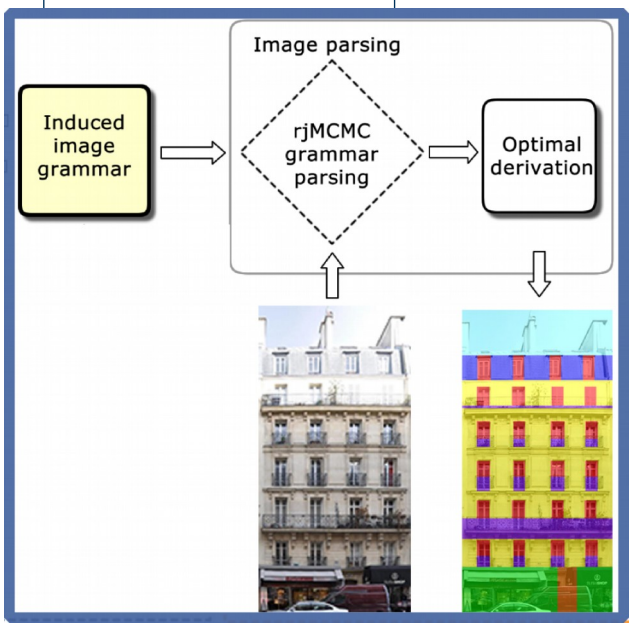
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



$$P(\delta|I) \propto \underbrace{P(I|\delta)}_{\text{Likelihood}} \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Rule term — Attribute term — Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

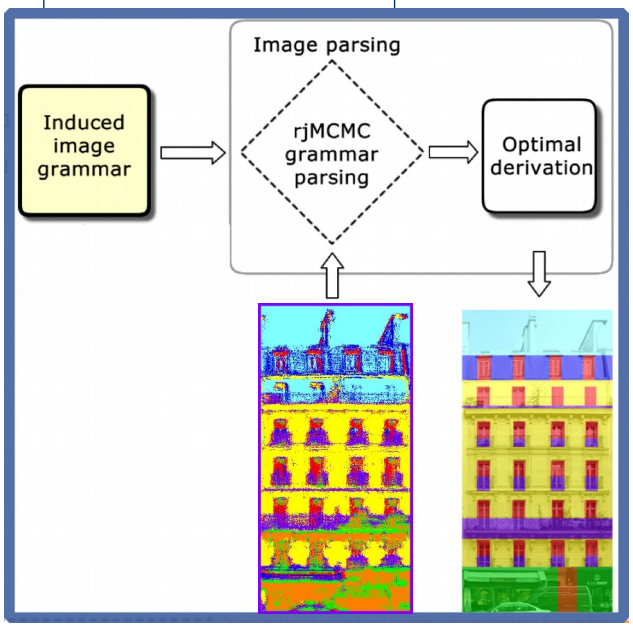
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space



Rule term Attribute term

$$P(\delta|I) \propto P(I|\delta) \underbrace{\prod_{s\in\delta} P(r_s) \prod_{s\in\delta} \phi(A(r_s))}_{P(\delta)}$$

Likelihood Prior

- Boltzmann's transformation:

$$E(\delta|I) = -log\, P(I|\delta) - \sum_{s\in\delta} log\, P(r_s) - \sum_{s\in\delta} log\, \phi(A(r_s))$$

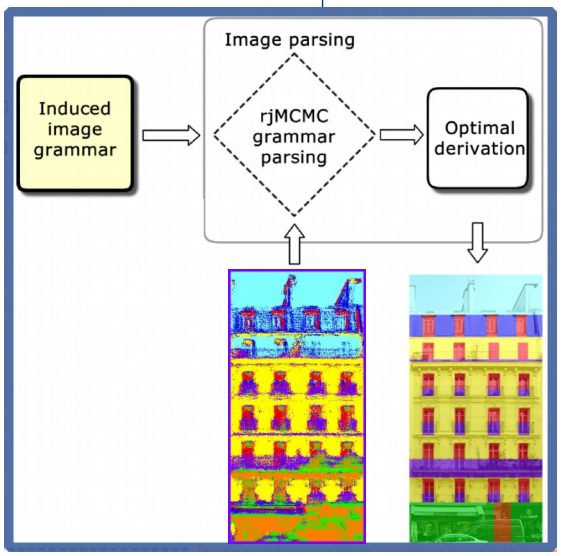
$$E_\delta = E_\delta^{image} + E_\delta^{rule} + E_\delta^{attribute}$$

- Rule term: sum of negative log probabilities of all rules in derivation δ
- Attribute term: discrepancy between proposed and expected attributes
- Image term: discrepancy between Random Forest pixel classifier and labeling induced by terminal nodes in δ

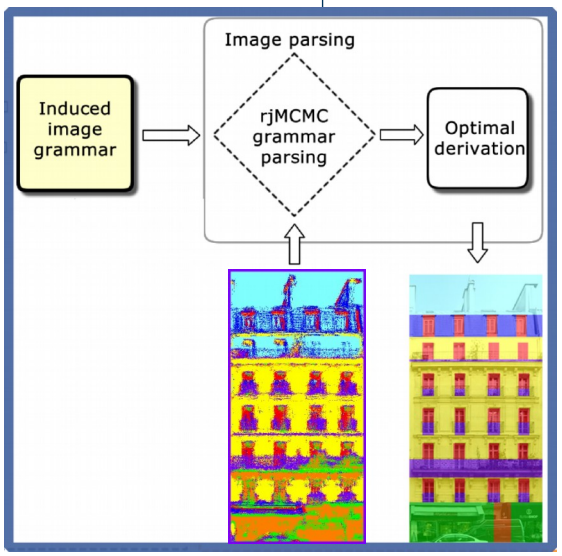
KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Parsing in image space - rjMCMC



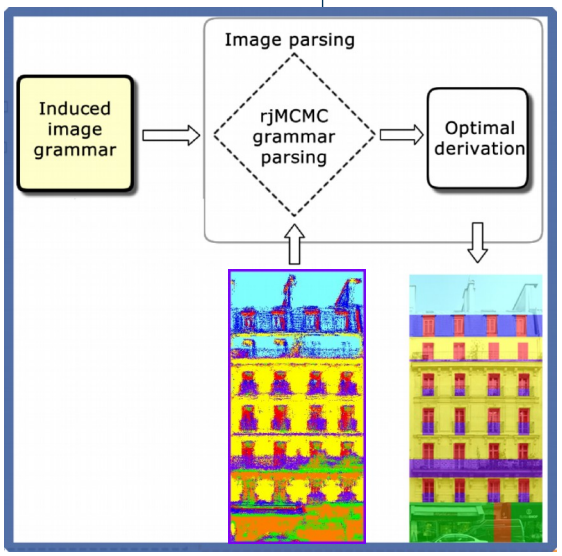
Parsing in image space - rjMCMC



- rjMCMC with Metropolis-Hastings update
- Markov chain is initialized with a random derivation

- Two possible moves in every iteration:
  - **Diffusion** move – parse tree doesn't change, dimensionality of α is preserved
  - **Jump** move – a rule in the tree is resampled, dimensionality of α changes
- Improvements:
  - **Parallel tempering** – 8 chains are run in parallel on different cooling schedules
  - **Delayed rejection –** a jump move is always followed by a diffusion move, and the moves are accepted or rejected in unison

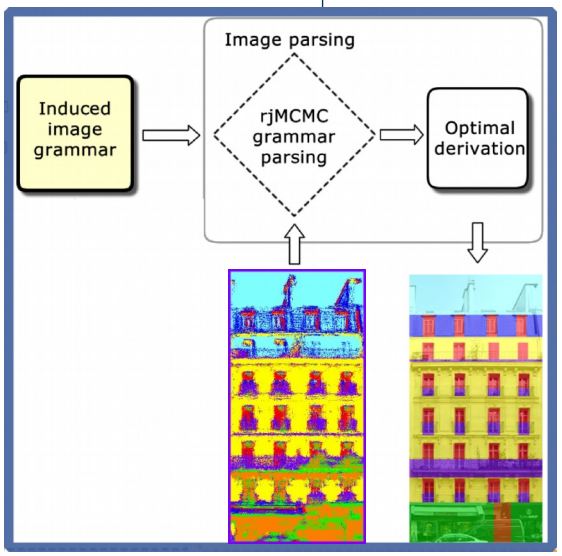
Parsing in image space - rjMCMC



- rjMCMC with Metropolis-Hastings update
- Markov chain is initialized with a random derivation

- Two possible moves in every iteration:
  - **Diffusion** move – parse tree doesn't change, dimensionality of α is preserved
  - **Jump** move – a rule in the tree is resampled, dimensionality of α changes
- Improvements:
  - **Parallel tempering** – 8 chains are run in parallel on different cooling schedules
  - **Delayed rejection –** a jump move is always followed by a diffusion move, and the moves are accepted or rejected in unison

Parsing in image space - rjMCMC



- rjMCMC with Metropolis-Hastings update
- Markov chain is initialized with a random derivation

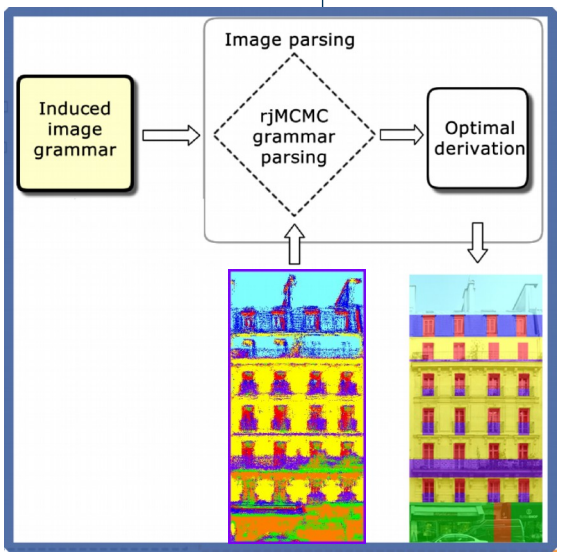
$$\delta = (\tau, \alpha)$$

Parse tree

"Parameter vector"
Concatenated attributes
in a pre-order traversal of τ

- Two possible moves in every iteration:
  - **Diffusion** move – parse tree doesn't change, dimensionality of α is preserved
  - **Jump** move – a rule in the tree is resampled, dimensionality of α changes
- Improvements:
  - **Parallel tempering** – 8 chains are run in parallel on different cooling schedules
  - **Delayed rejection –** a jump move is always followed by a diffusion move, and the moves are accepted or rejected in unison

Parsing in image space - rjMCMC



- rjMCMC with Metropolis-Hastings update
- Markov chain is initialized with a random derivation

$$\delta = (\tau, \alpha)$$

Parse tree

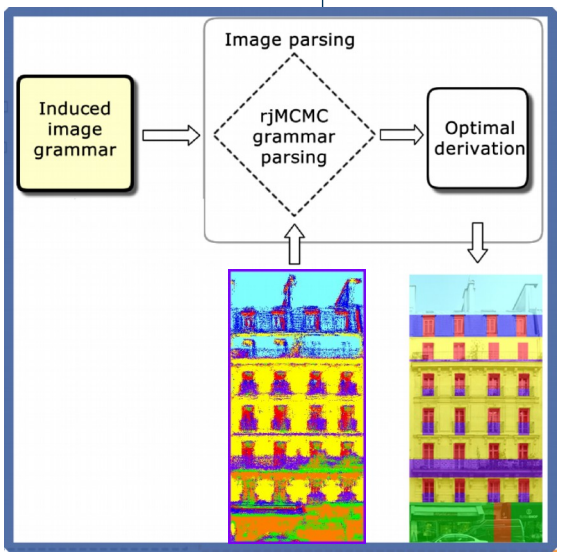
"Parameter vector"
Concatenated attributes
in a pre-order traversal of τ

- Two possible moves in every iteration:
  - **Diffusion** move – parse tree doesn't change, dimensionality of α is preserved
  - **Jump** move – a rule in the tree is resampled, dimensionality of α changes
- Improvements:
  - **Parallel tempering** – 8 chains are run in parallel on different cooling schedules
  - **Delayed rejection –** a jump move is always followed by a diffusion move, and the moves are accepted or rejected in unison

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing in image space - rjMCMC



- rjMCMC with Metropolis-Hastings update
- Markov chain is initialized with a random derivation

$$\delta = (\tau, \alpha)$$

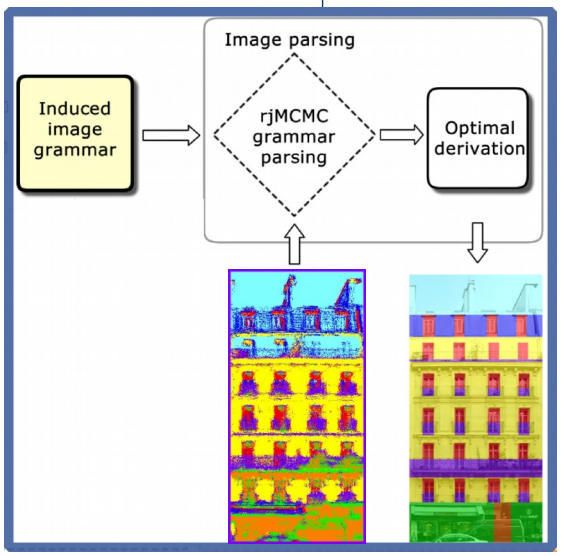
Parse tree

"Parameter vector"
Concatenated attributes in a pre-order traversal of τ

- Two possible moves in every iteration:
  - **Diffusion** move – parse tree doesn't change, dimensionality of α is preserved
  - **Jump** move – a rule in the tree is resampled, dimensionality of α changes
- Improvements:
  - **Parallel tempering** – 8 chains are run in parallel on different cooling schedules
  - **Delayed rejection –** a jump move is always followed by a diffusion move, and the moves are accepted or rejected in unison

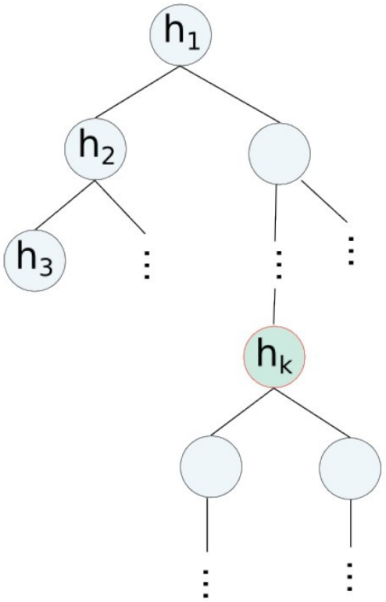
rjMCMC diffusion move

- Select a random node h_k in the derivation tree
- Sample from a Gaussian proposal distribution centered on the current parameters



$\delta=(\tau,\alpha)$

τ:



Starting α:

$[\underbrace{\alpha_1\alpha_2\ldots\alpha_{n_1}}_{h_1}\ \underbrace{\alpha_{n_1+1}\ldots\alpha_{n_2}}_{h_2}\ \ldots\ \underbrace{\alpha_{n_{k-1}}\ldots\alpha_{n_k}}_{h_k}\ \ldots]$

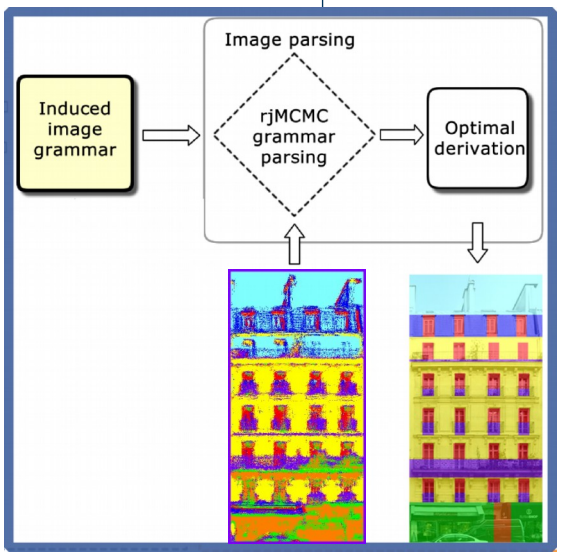
Diffusion move to α':

$[\underbrace{\alpha_1\alpha_2\ldots\alpha_{n_1}}_{h_1}\ \underbrace{\alpha_{n_1+1}\ldots\alpha_{n_2}}_{h_2}\ \ldots\ \underbrace{\alpha'_{n_{k-1}}\ldots\alpha'_{n_k}}_{h_k}\ \ldots]$

Acceptance probability of the move

$$\rho_{\delta\to\delta'} = min\{1, \frac{p(\delta'|I)}{p(\delta|I)}\} = min\{1, e^{-(E_{\delta'}-E_\delta)}\}$$

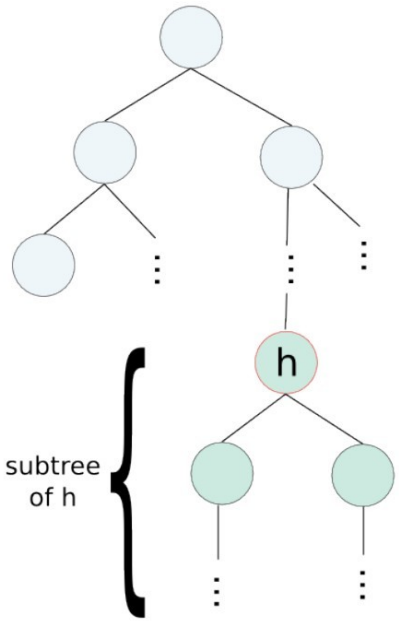
rjMCMC jump move



- A random node h is selected in the derivation tree
- A new rule is sampled from all rules applicable to selected LHS
- If RHS has different length, the whole subtree rooted on h is re-derived
  - Topology of τ changes
  - Dimensionality of α changes

$\delta=(\tau,\alpha)$

τ:



Starting α:

$[\alpha_1\alpha_2...\alpha_l \underbrace{\alpha_{l+1}...\alpha_{l+d_1}}_{\text{in subtree of h}} \alpha_{l+d_1+1}...\alpha_n]$

Jump move to α':

$u = [u_1..u_{d_2}]$

$\alpha' = [\alpha'_1\alpha'_2...\alpha'_l \alpha'_{l+1}...\alpha'_{l+d_2} \alpha'_{l+d_2+1}...\alpha'_m]$

$= [\alpha_1\alpha_2...\alpha_l \quad u_1..u_{d_2} \quad \alpha_{l+d_1+1}...\alpha_n]$

$u' = [\alpha'_{l+1}...\alpha'_{l+d_2}]$

Acceptance probability of the move

$$\rho_{\delta\to\delta'} = min\{1, \frac{q_{\tau'}(h)}{q_\tau(h)} e^{-[(E^{img}_{\delta'}+E^{attr}_{\delta'})-(E^{img}_{\delta}+E^{attr}_{\delta})]}\}$$

Probability of choosing a nonterminal h in tree τ

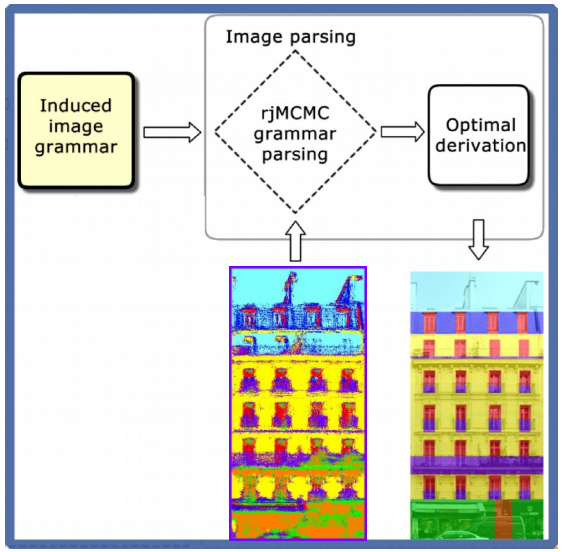
Image and attribute terms

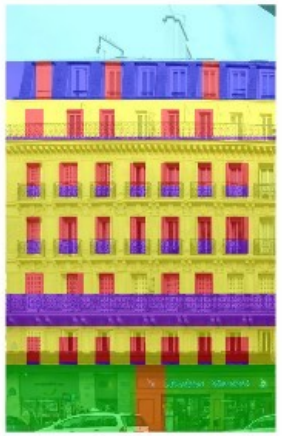
KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Parsing results

- ECP dataset
- Example: fold 1

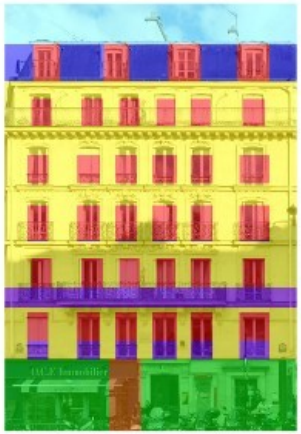



monge_1


monge_3


monge_6

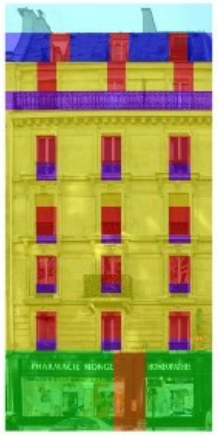

monge_9


monge_29bis


monge_61

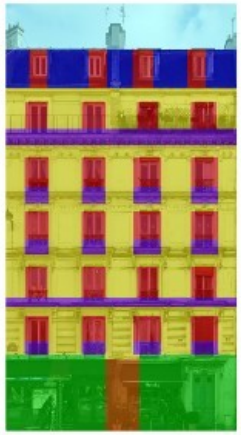

monge_65


monge_71

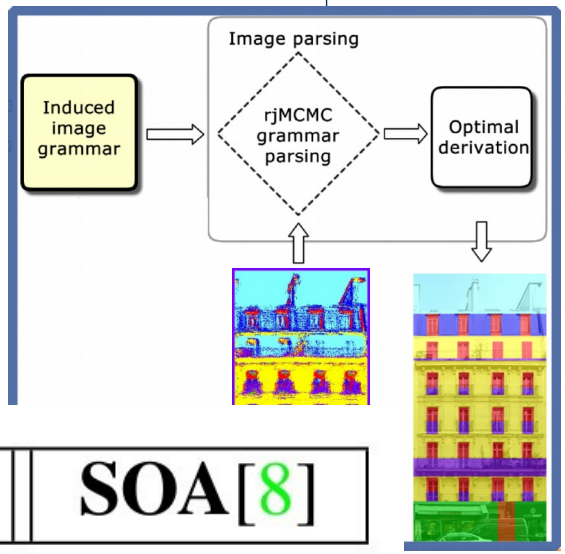

monge_74


monge_79bis


monge_86


monge_101

Parsing results



- Quantitative analysis

Class	RF[22]	RL[21]	Ours	SOA[8]
Window	29	62	66	75
Wall	58	82	80	88
Balcony	35	58	49	70
Door	79	47	50	67
Roof	51	66	71	74
Sky	73	95	91	97
Shop	20	88	81	93
Overall	48.55	74.71	74.82	84.17

KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Generating novel designs

KU LEUVEN

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Generating novel designs

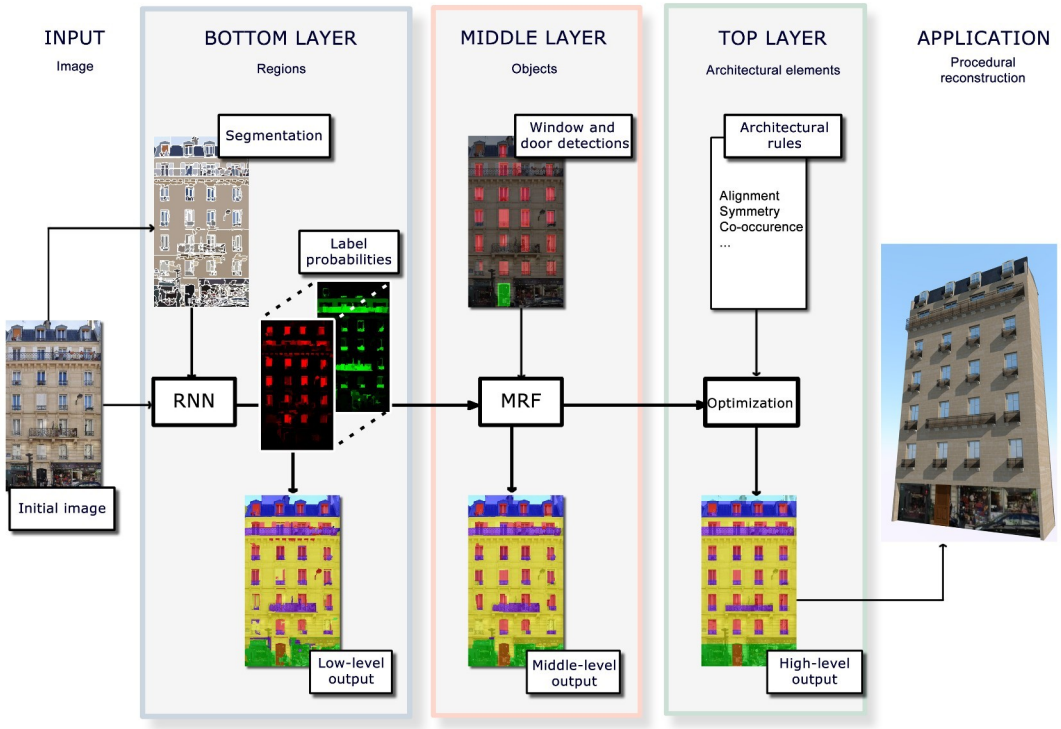
- The effect of prior weight *w* on grammar generalization

w=1.0

w=0.3

Future work

- State of the art in facade parsing
  - Images → Labelings
- This paper
  - Labelings → Grammar
- Combine the two approaches
  - “Facade bootstrapping”



A. Martinović, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In ECCV, 2012

KU LEUVEN

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISICS

CVL Computer Vision Lab

Thank you