Cube Summing, Approximate Inference with Non-Local Features, and Dynamic Programming without Semirings

Kevin Gimpel and Noah A. Smith

Overview

- We introduce **cube summing,** which extends dynamic programming algorithms for summing with non-local features
  - Inspired by **cube pruning** (Chiang, 2007; Huang & Chiang, 2007)
- We relate cube summing to semiring-weighted logic programming
  - Without non-local features, cube summing is a novel semiring
  - Non-local features break some of the semiring properties
  - We propose an implementation based on **arithmetic circuits**

Outline

- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion

Fundamental Problems

- Consider an exponential probabilistic model

$$p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$$

- Two fundamental problems we often need to solve
  - Decoding

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

  - Summing

$$s(x) = \sum_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

Fundamental Problems

- Consider an exponential probabilistic model

$$p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$$

example: HMM

x is a sentence, y is a tag sequence

- Two fundamental problems we often need to solve
  - Decoding

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

Viterbi algorithm

  - Summing

$$s(x) = \sum_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

forward and backward algorithms

Fundamental Problems

- Consider an exponential probabilistic model

$$p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$$

example: PCFG

x is a sentence, y is a parse tree

- Two fundamental problems we often need to solve
  - Decoding

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \prod_{m=1}^{M} \lambda_m{}^{h_m(x,y)}$$

probabilistic CKY

  - Summing

$$s(x) = \sum_{y \in \mathcal{Y}} \prod_{m=1}^{M} \lambda_m{}^{h_m(x,y)}$$

inside algorithm

ARK lti Carnegie Mellon

Fundamental Problems

- Consider an exponential probabilistic model

$$p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$$

- Two fundamental problems we often need to solve
  - Decoding

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

  - Summing

$$s(x) = \sum_{y \in \mathcal{Y}} \prod_{m=1}^{M} {\lambda_m}^{h_m(x,y)}$$

	supervised:	*unsupervised:*
Decoding	*perceptron, MIRA, MERT*	*self-training, Viterbi EM*
Summing	*log-linear models*	*EM, hidden-variable models*

Carnegie Mellon

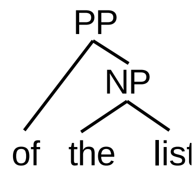
Dynamic Programming

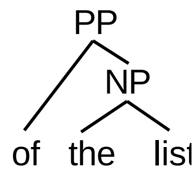
- Consider the probabilistic CKY algorithm

$$C_{X,i-1,i} = \lambda_{X \rightarrow w_i}$$

$$C_{X,i,k} = \max_{Y,Z \in \mathcal{N}; j \in \{i+1,\ldots,k-1\}} \lambda_{X \rightarrow Y Z} \times C_{Y,i,j} \times C_{Z,j,k}$$

$$goal = C_{S,0,n}$$

Weighted Logic Programs	Probabilistic CKY	Example
theorem	chart item	$C_{X,i,j}$
axiom	rule probability	$\lambda_{X \to Y\,Z}$
proof	derivation	

Weighted Logic Programs	Probabilistic CKY	Example
theorem	chart item	$C_{X,i,j}$
axiom	rule probability	$\lambda_{X \rightarrow Y\,Z}$
proof	derivation	

- In semiring-weighted logic programming, theorem and axiom values come from a **semiring**

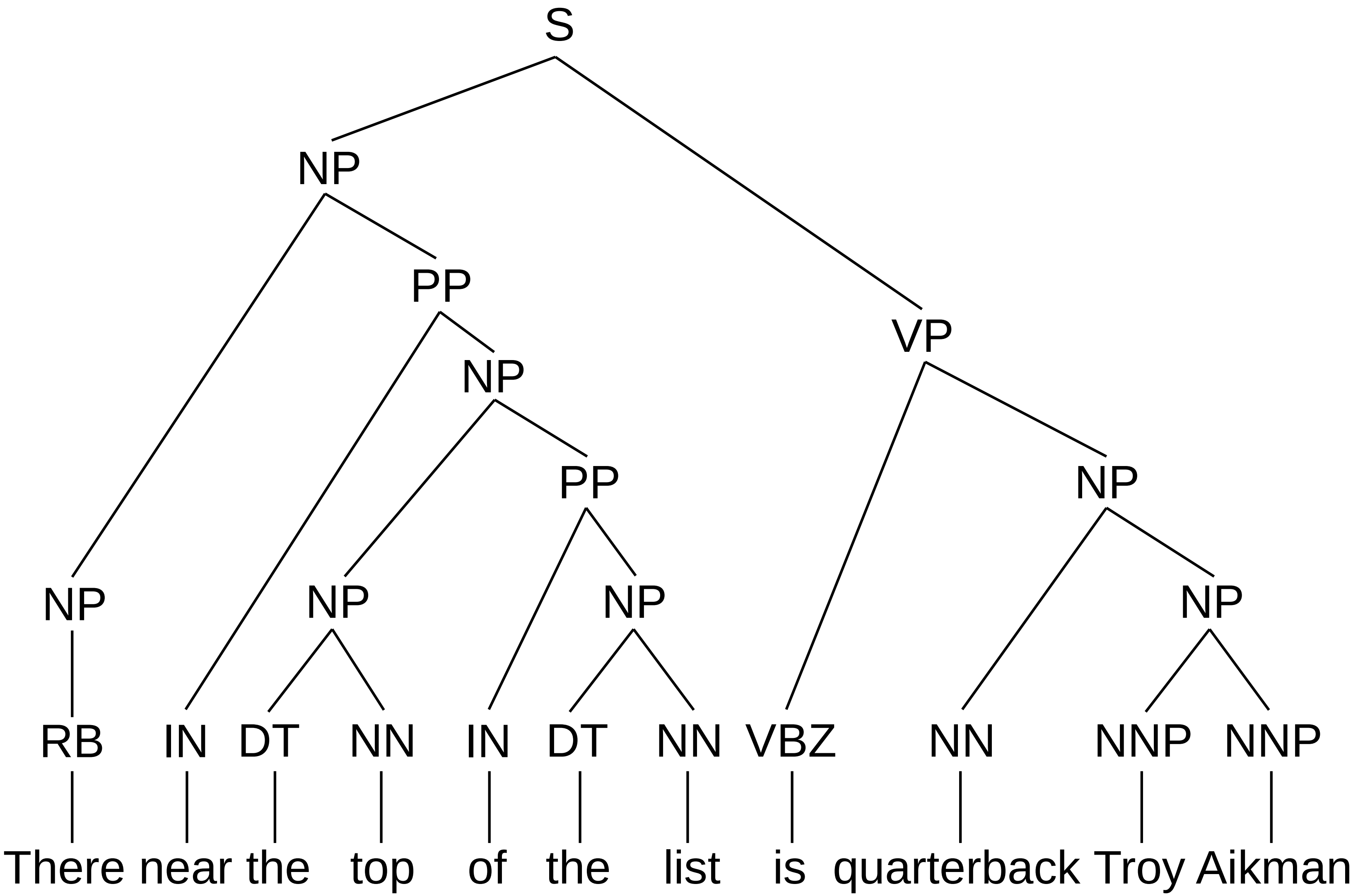
Features

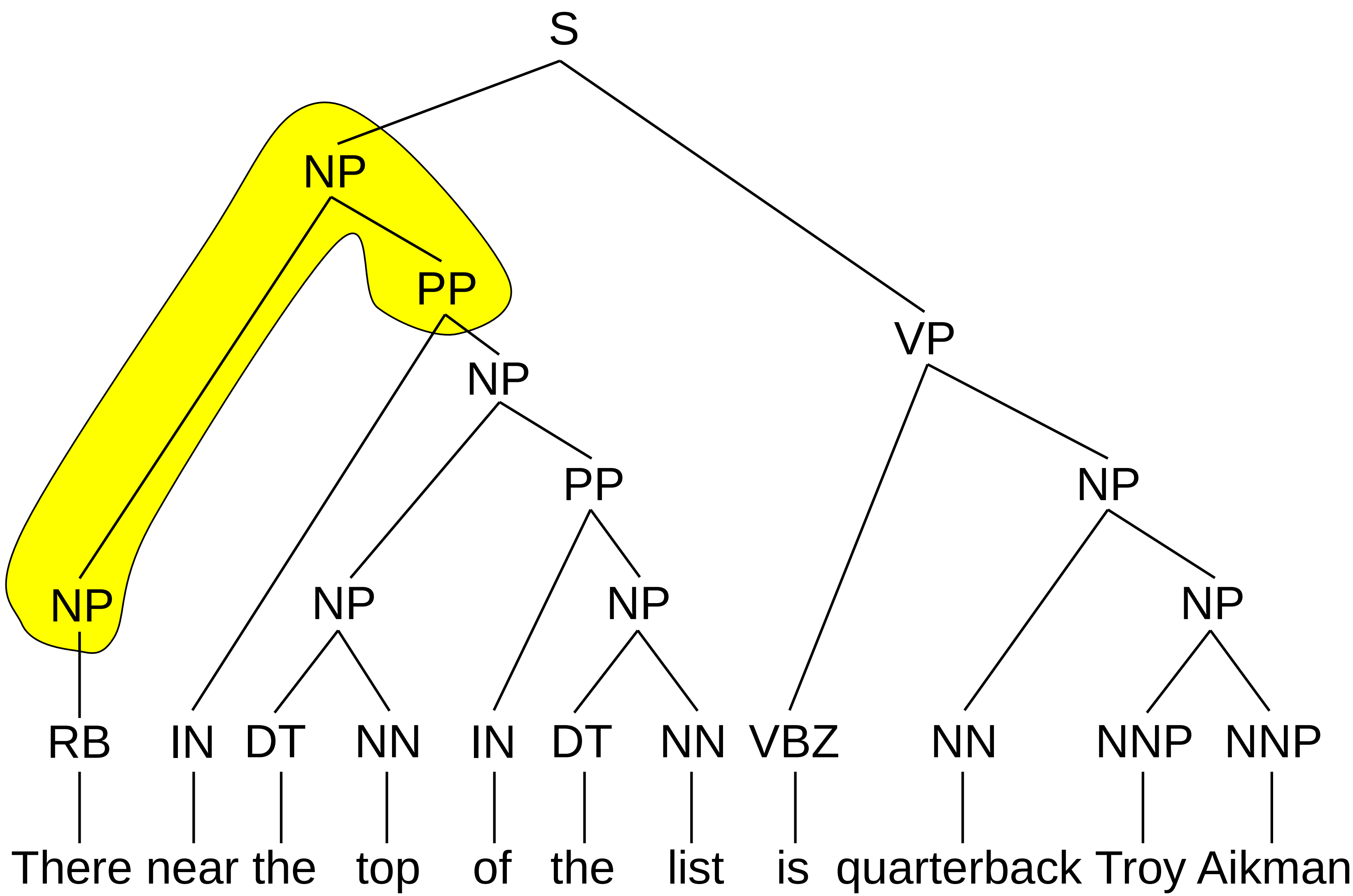
- Recall our model: $p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$
- The $h_m(x, y)$ are feature functions and the λ_m are nonnegative weights

Features

- Recall our model: $p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$
- The $h_m(x, y)$ are feature functions and the λ_m are nonnegative weights
- **Local features** depend *only* on theorems used in an equation (or any of the axioms), *not* on the proofs of those theorems

$$C_{X,i,k} = \max_{Y,Z \in \mathcal{N}; j \in \{i+1,\ldots,k-1\}} \lambda_{X \to Y Z} \times C_{Y,i,j} \times C_{Z,j,k}$$


S
NP
VP
PP
NP
NP
PP
NP
NP
NP
NP
RB IN DT NN IN DT NN VBZ NN NNP NNP
There near the top of the list is quarterback Troy Aikman


S
NP
PP
NP
VP
PP
NP
NP
NP
NP
NP
RB IN DT NN IN DT NN VBZ NN NNP NNP
There near the top of the list is quarterback Troy Aikman

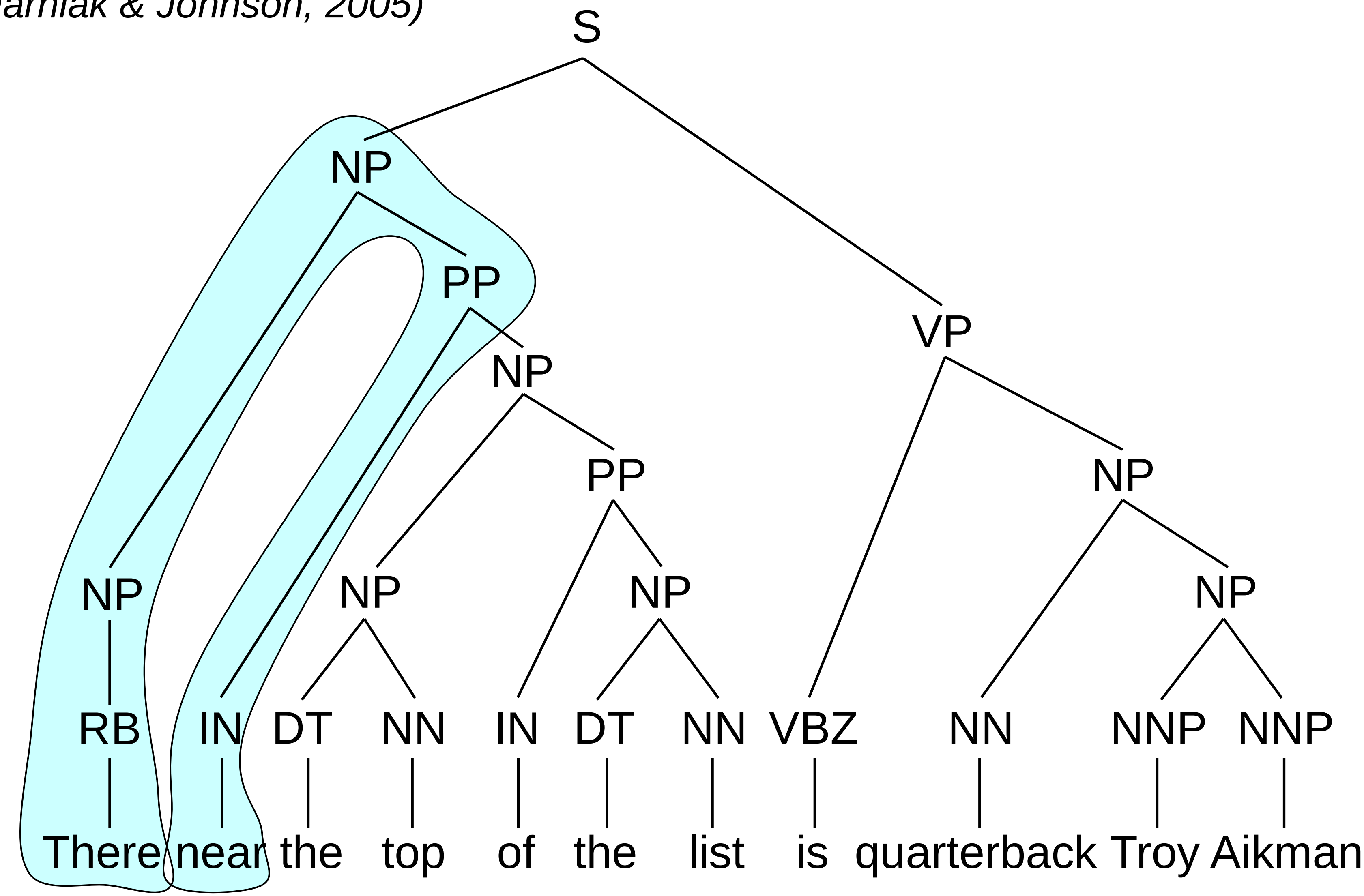
Features

- Recall our model: $p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)}$
- The $h_m(x,y)$ are feature functions and the λ_m are nonnegative weights
- **Local features** depend *only* on theorems used in an equation (or any of the axioms), *not* on the proofs of those theorems

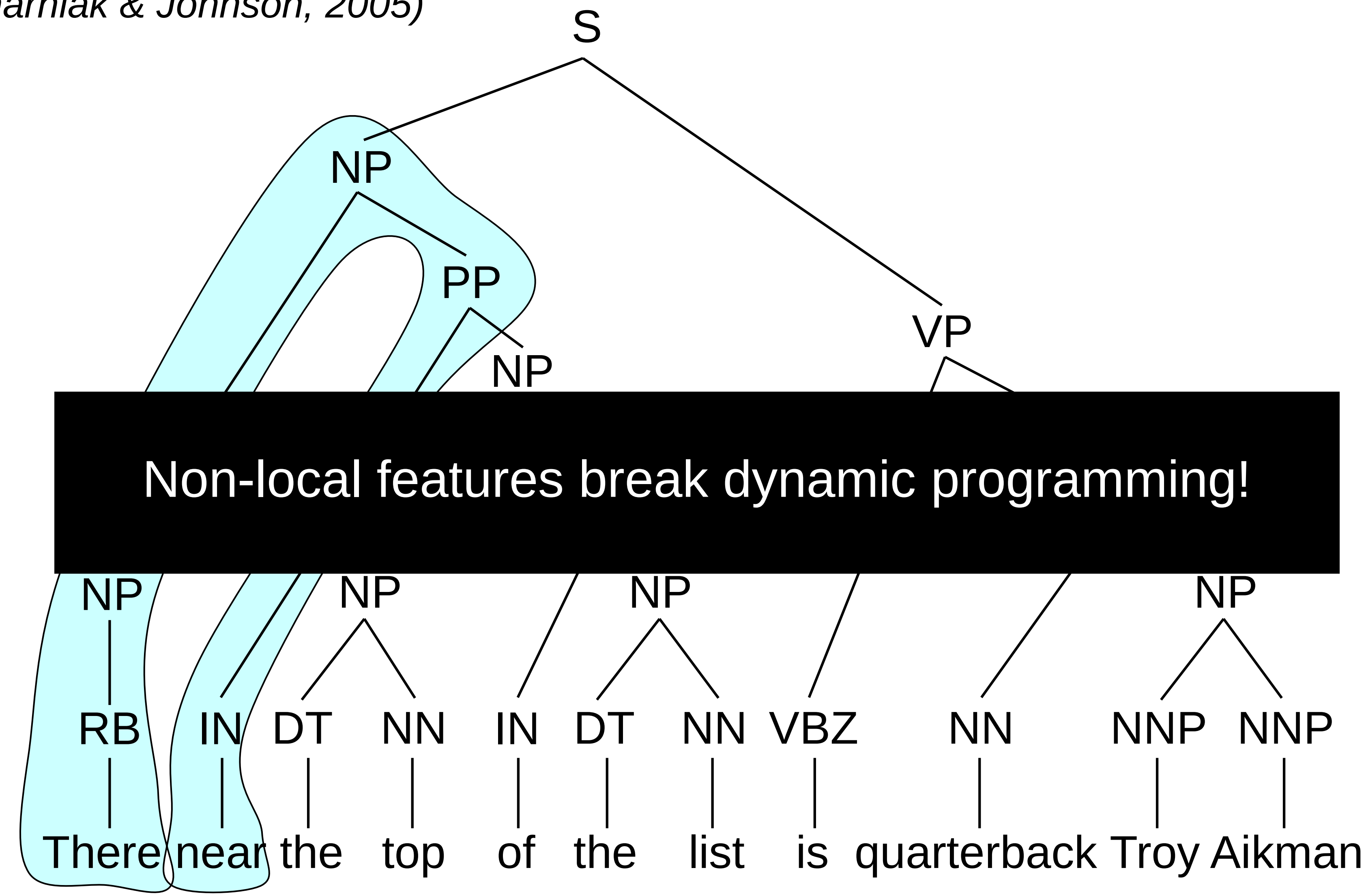
$$C_{X,i,k} = \max_{Y,Z \in \mathcal{N}; j \in \{i+1,\ldots,k-1\}} \lambda_{X \to Y Z} \times C_{Y,i,j} \times C_{Z,j,k}$$

- **Non-local features** depend on theorem proofs

“NGramTree” feature
(Charniak & Johnson, 2005)



"NGramTree" feature
(Charniak & Johnson, 2005)



Non-local features break dynamic programming!

Other Algorithms for Approximate Inference

- Beam search (Lowerre, 1979)
- Reranking (Collins, 2000)
- Algorithms for graphical models
  - Variational methods (MacKay, 1997; Beal, 2003; Kurihara & Sato, 2006)
  - Belief propagation (Sutton & McCallum, 2004; Smith & Eisner, 2008)
  - MCMC (Finkel et al., 2005; Johnson et al., 2007)
  - Particle filtering (Levy et al., 2009)
- Integer linear programming (Roth & Yih, 2004)
- Stacked learning (Cohen & Carvalho, 2005; Martins et al., 2008)
- Cube pruning (Chiang, 2007; Huang & Chiang, 2007)

Other Algorithms for Approximate Inference

- Beam search (Lowerre, 1979)
- Reranking (Collins, 2000)
- Algorithms for graphical models
  - Variational methods (MacKay, 1997; Beal, 2003; Kurihara & Sato, 2006)
  - Belief propagation (Sutton & McCallum, 2004; Smith & Eisner, 2008)
  - MCMC (Finkel et al., 2005; Johnson et al., 2007)
  - Particle filtering (Levy et al., 2009)
- Integer linear programming (Roth & Yih, 2004)
- Stacked learning (Cohen & Carvalho, 2005; Martins et al., 2008)
- Cube pruning (Chiang, 2007; Huang & Chiang, 2007)

- Why add one more?
  - Cube pruning extends **existing, widely-understood** dynamic programming algorithms for **decoding**
  - We want this for **summing** too

Outline

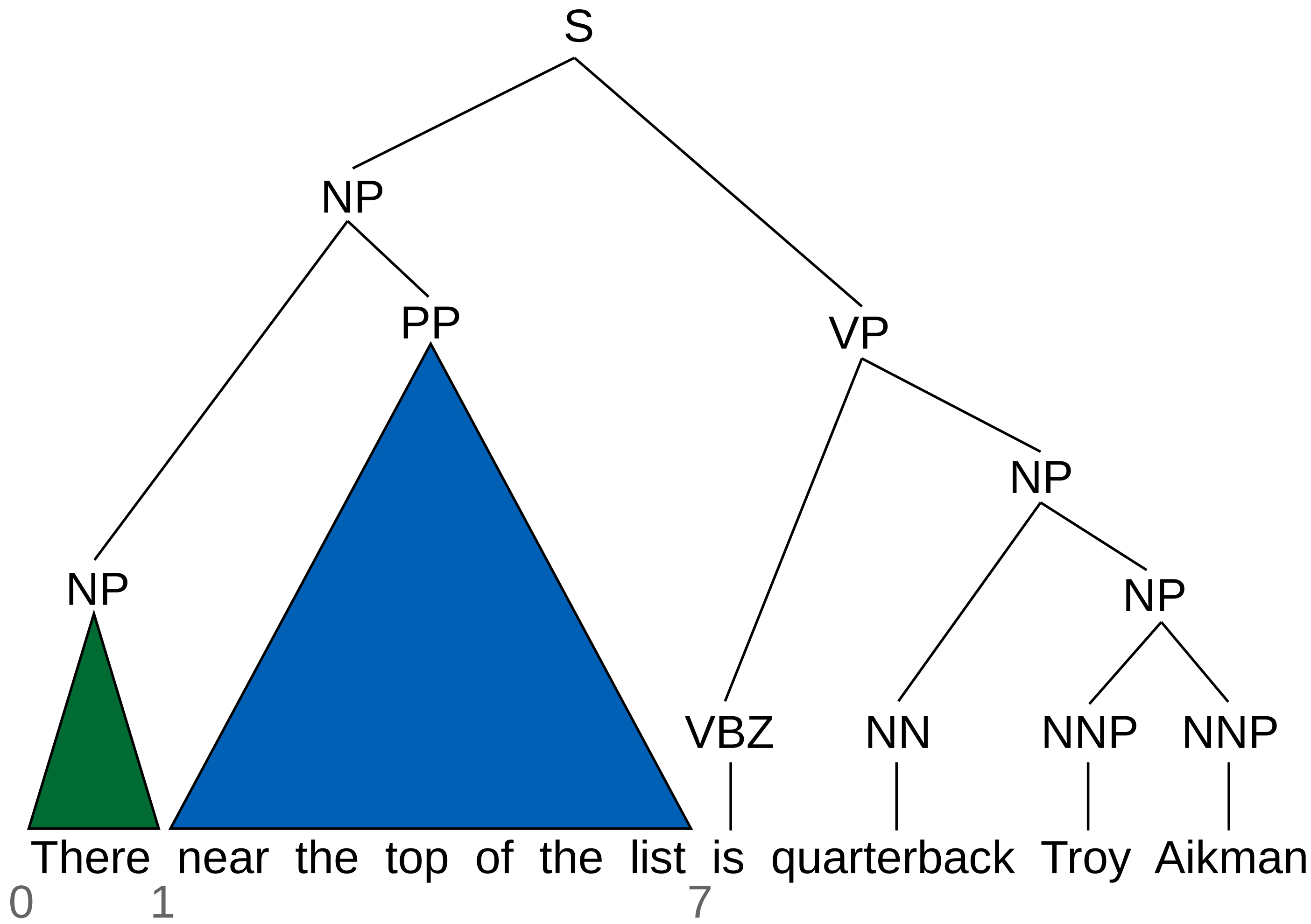
- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion

Cube Pruning

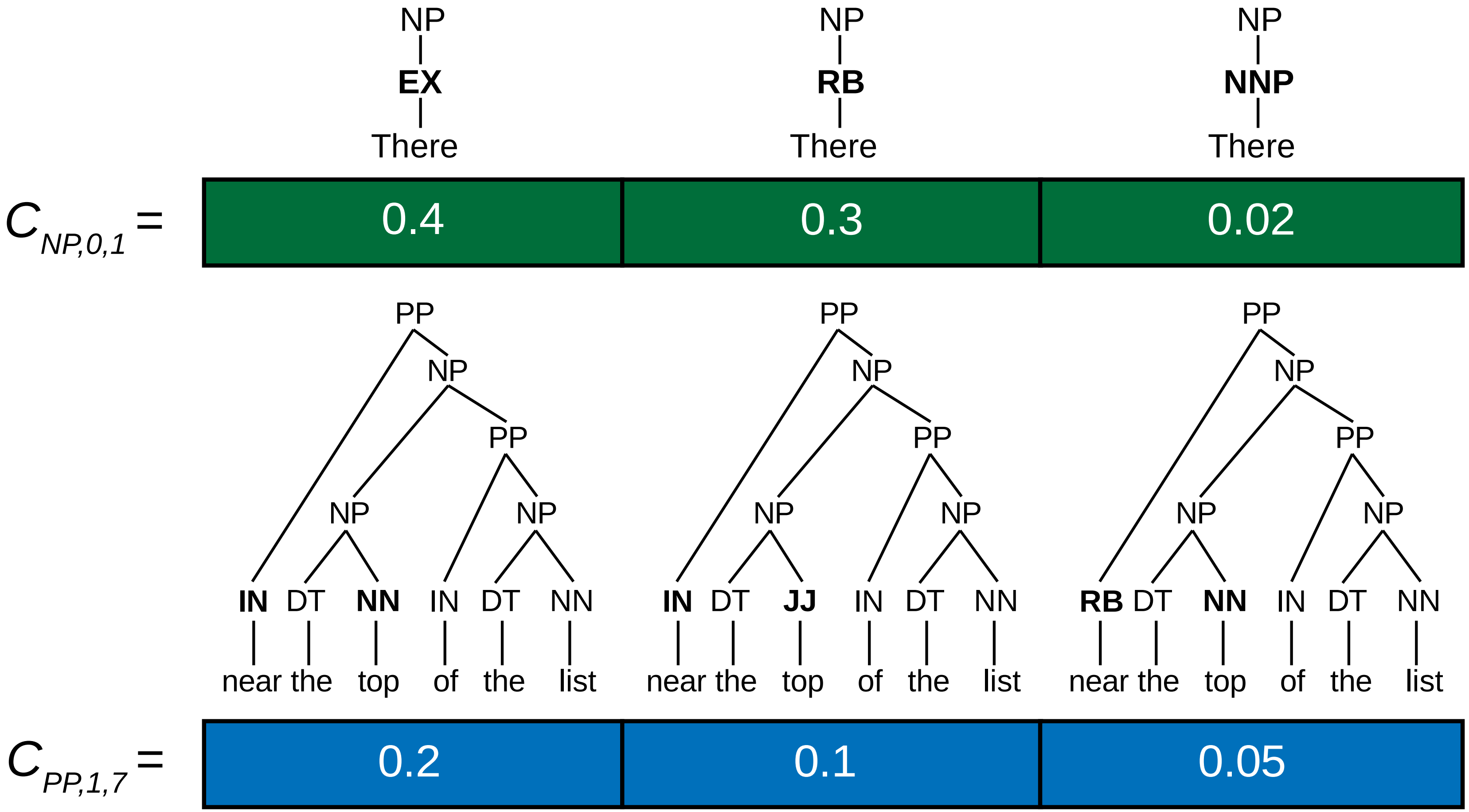
(Chiang, 2007; Huang & Chiang, 2007)

- Modification to dynamic programming algorithms for decoding to use non-local features approximately
- Keeps a k-best list of proofs for each theorem
- Applies non-local feature functions on these proofs when proving new theorems

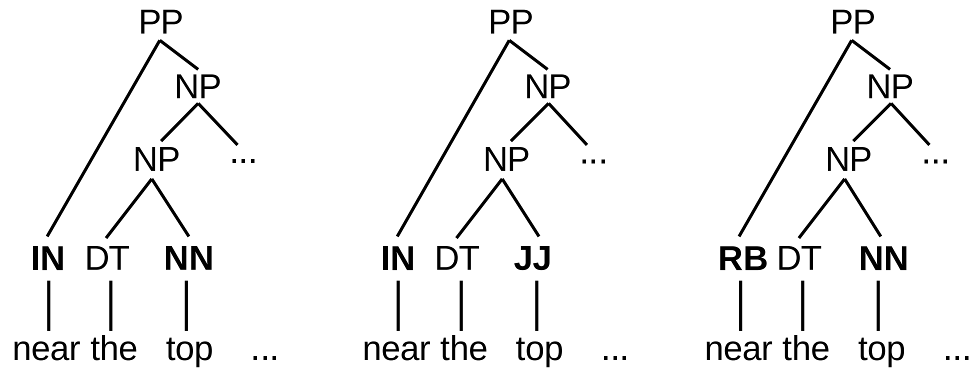
$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$

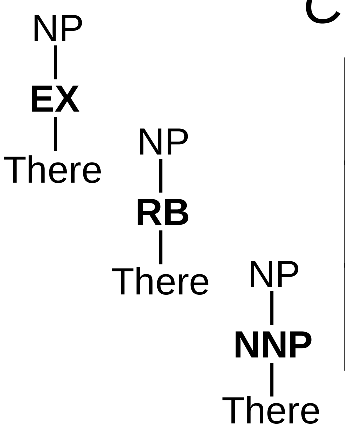


$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$



$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$

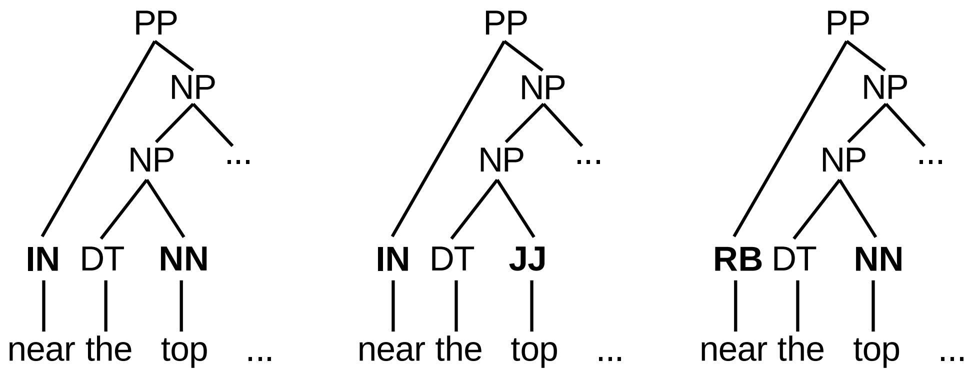


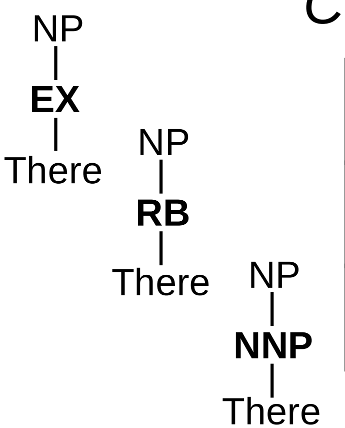


$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.08	0.04	0.02
0.3	0.06	0.03	0.015
0.02	0.004	0.002	0.001

$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$

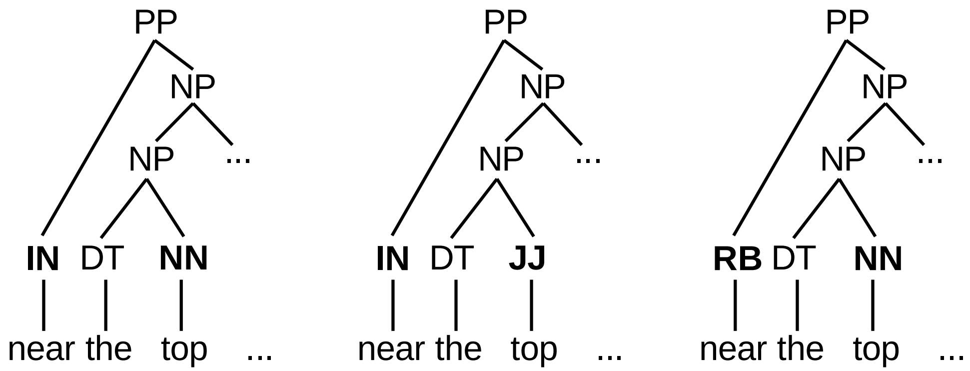
$$\lambda_{NP \rightarrow NP\ PP} = 0.5$$

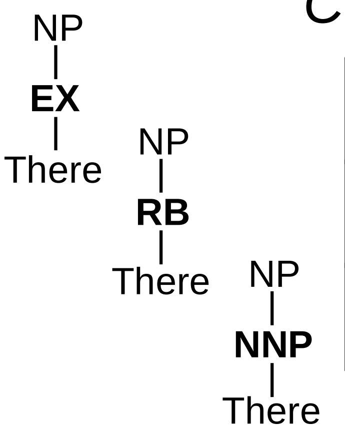




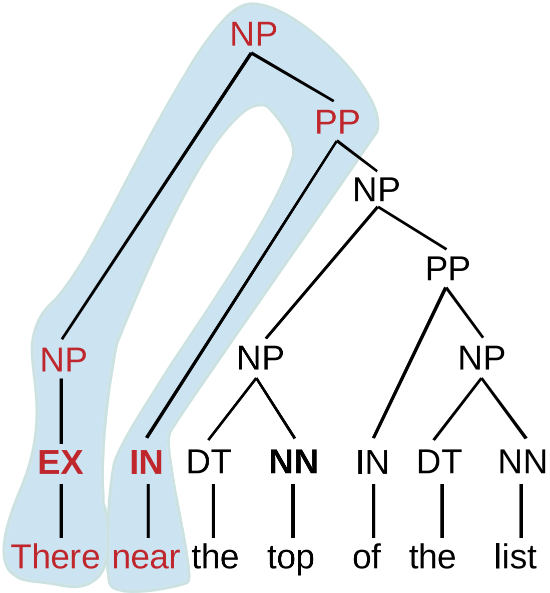
$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.08 × 0.5	0.04 × 0.5	0.02 × 0.5
0.3	0.06 × 0.5	0.03 × 0.5	0.015 × 0.5
0.02	0.004 × 0.5	0.002 × 0.5	0.001 × 0.5

$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$

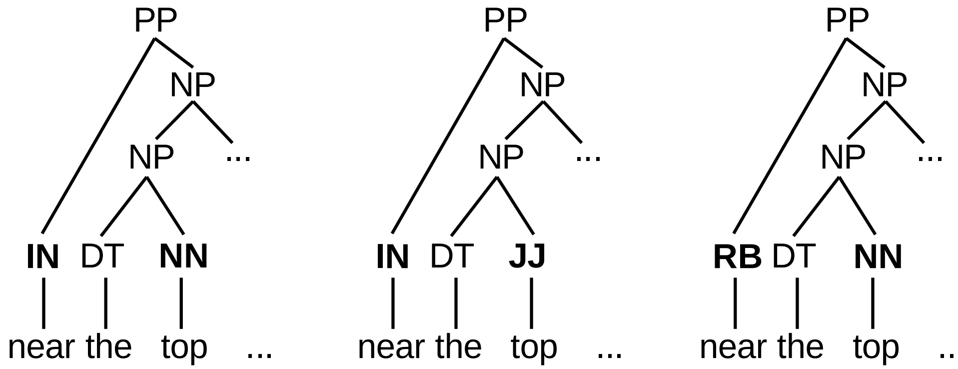


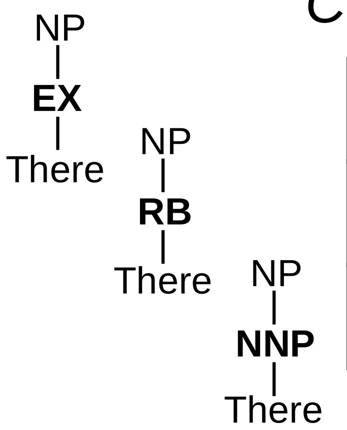


$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.04	0.02	0.01
0.3	0.03	0.015	0.0075
0.02	0.002	0.001	0.0005



$$\lambda_{There\ EX\ NP\ NP\ PP\ IN\ near} = 0.2$$





$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.04 × 0.2	0.02 × 0.2	0.01
0.3	0.03	0.015	0.0075
0.02	0.002	0.001	0.0005

$\lambda_{There\ EX\ NP\ NP\ PP\ IN\ near} = 0.2$

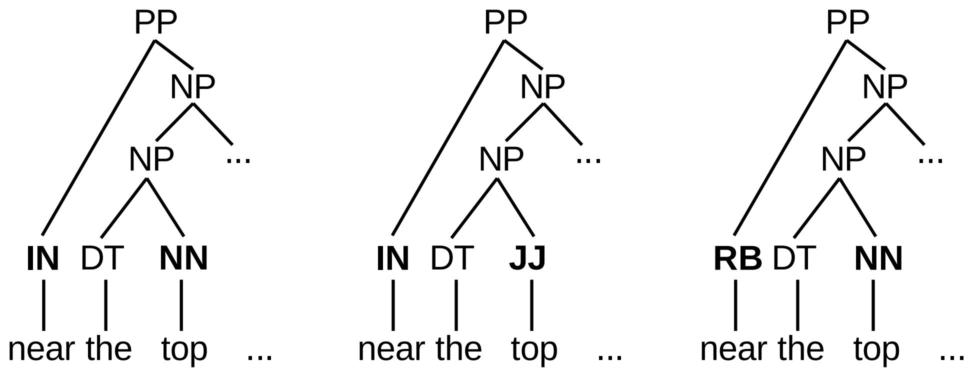
$\lambda_{There\ RB\ NP\ NP\ PP\ IN\ near} = 0.6$

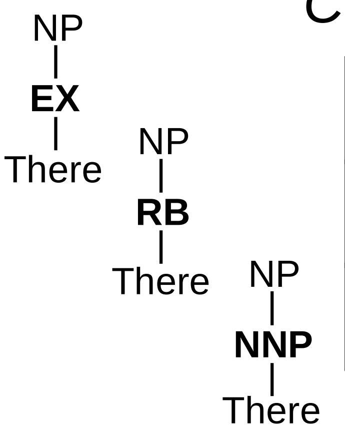
$\lambda_{There\ NNP\ NP\ NP\ PP\ IN\ near} = 0.1$

$\lambda_{There\ EX\ NP\ NP\ PP\ RB\ near} = 0.1$

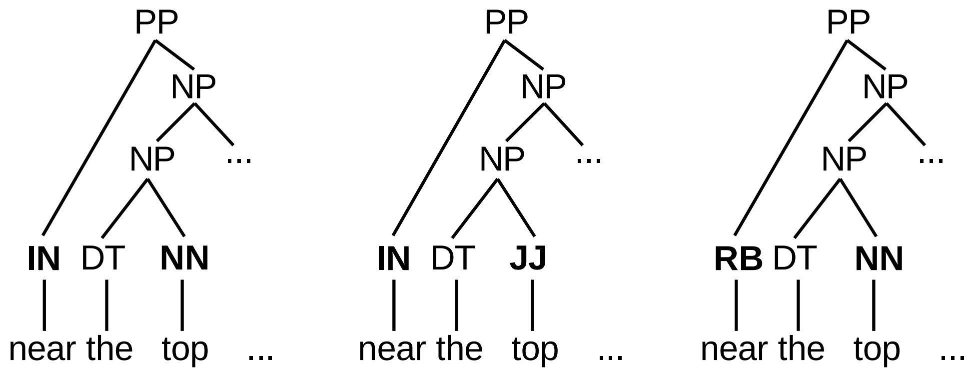
$\lambda_{There\ RB\ NP\ NP\ PP\ RB\ near} = 0.4$

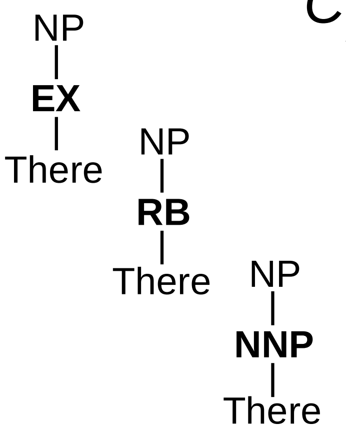
$\lambda_{There\ NNP\ NP\ NP\ PP\ RB\ near} = 0.2$



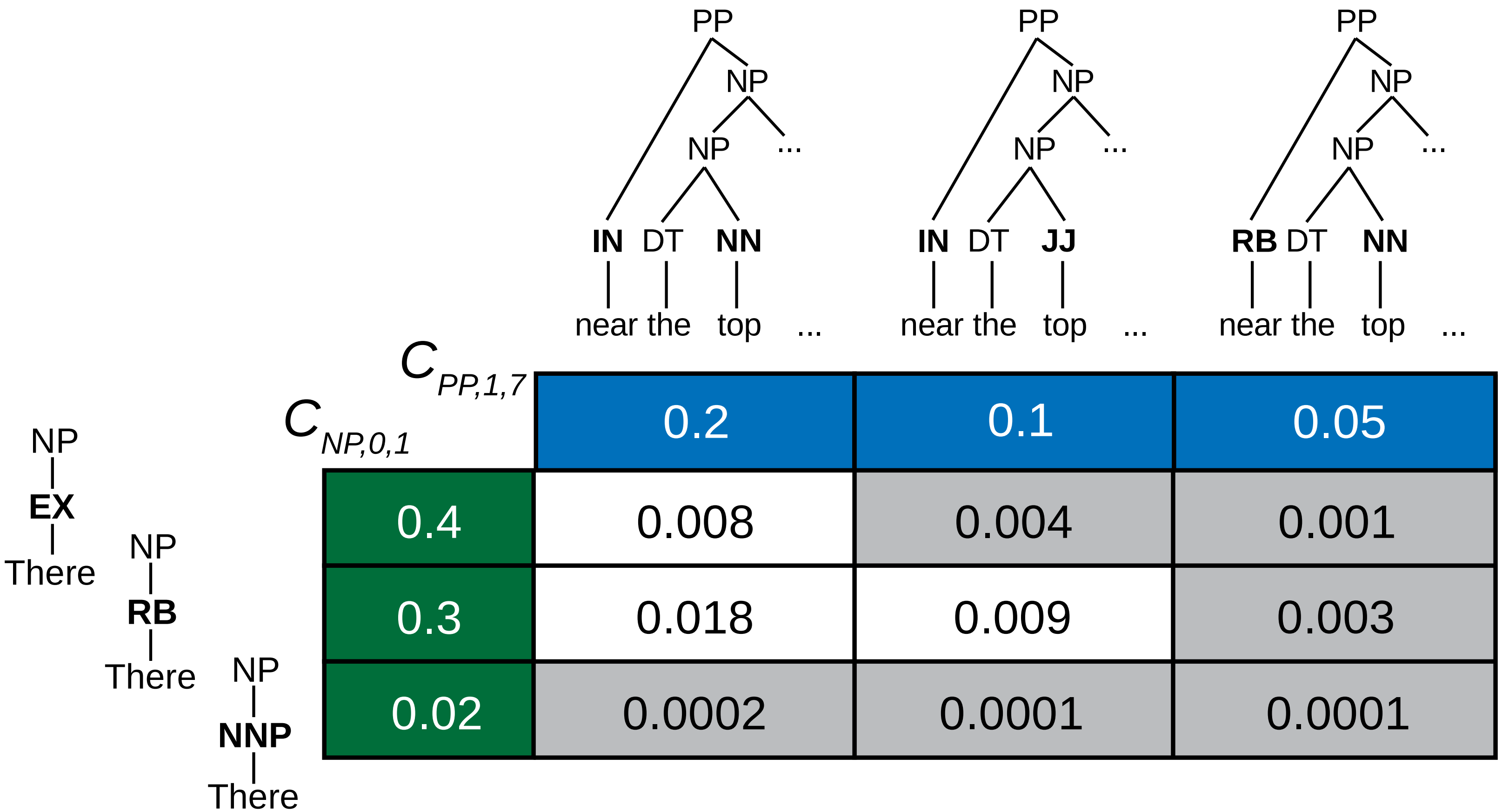


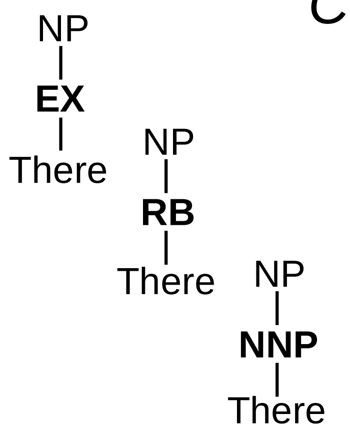
$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.04 × 0.2	0.02 × 0.2	0.01 × 0.1
0.3	0.03 × 0.6	0.015 × 0.6	0.0075 × 0.4
0.02	0.002 × 0.1	0.001 × 0.1	0.0005 × 0.2



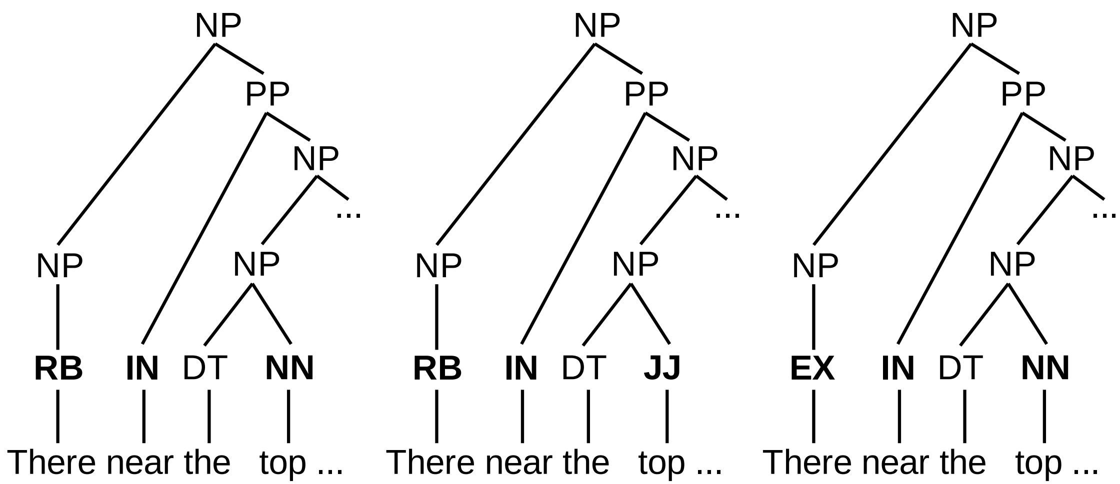


$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.008	0.004	0.001
0.3	0.018	0.009	0.003
0.02	0.0002	0.0001	0.0001


PP
NP
NP
...
IN DT NN
near the top ...
PP
NP
NP
...
IN DT JJ
near the top ...
PP
NP
NP
...
RB DT NN
near the top ...
C PP,1,7
C NP,0,1
0.2
0.1
0.05
NP
EX
There
NP
RB
There
NP
NNP
There
0.4
0.008
0.004
0.001
0.3
0.018
0.009
0.003
0.02
0.0002
0.0001
0.0001



$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05
0.4	0.008	0.004	0.001
0.3	0.018	0.009	0.003
0.02	0.0002	0.0001	0.0001



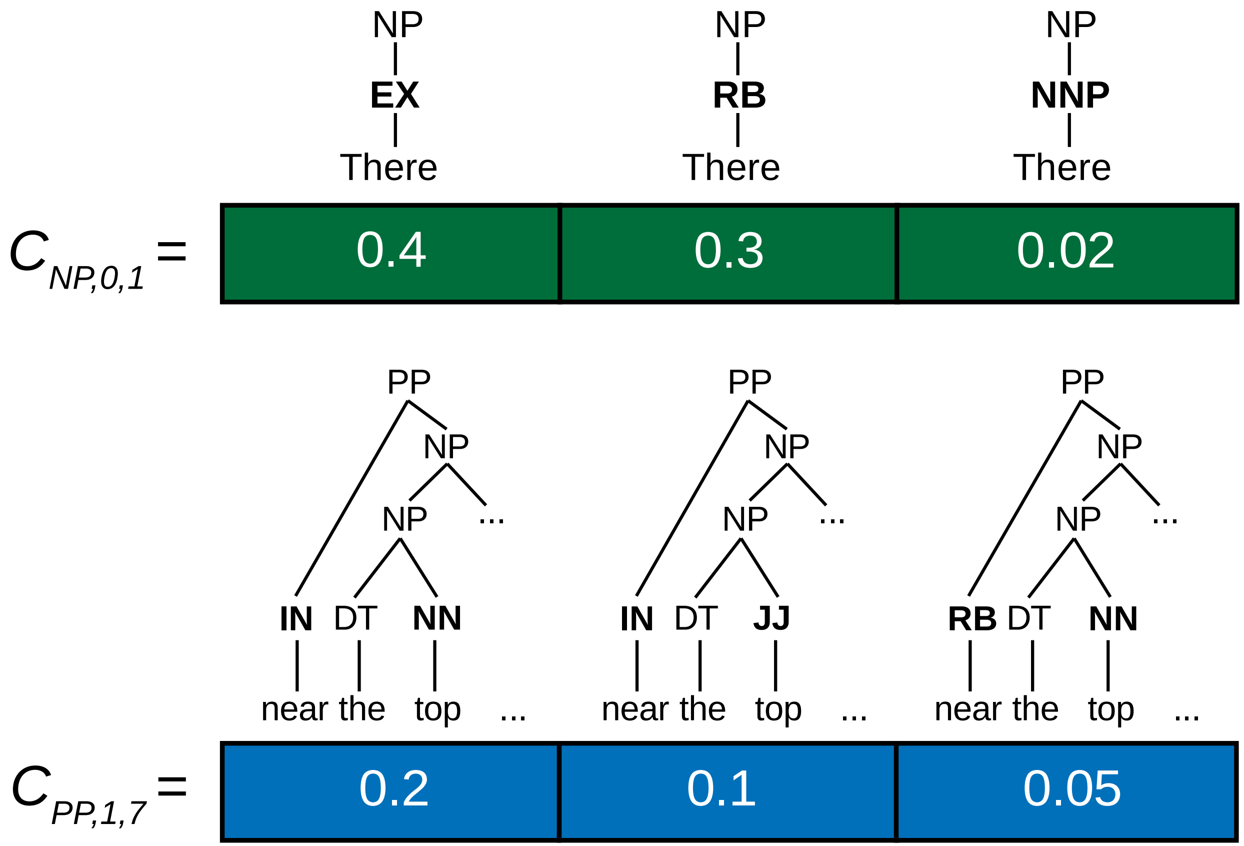
$C_{NP,0,7}$	0.018	0.009	0.008

Clarification

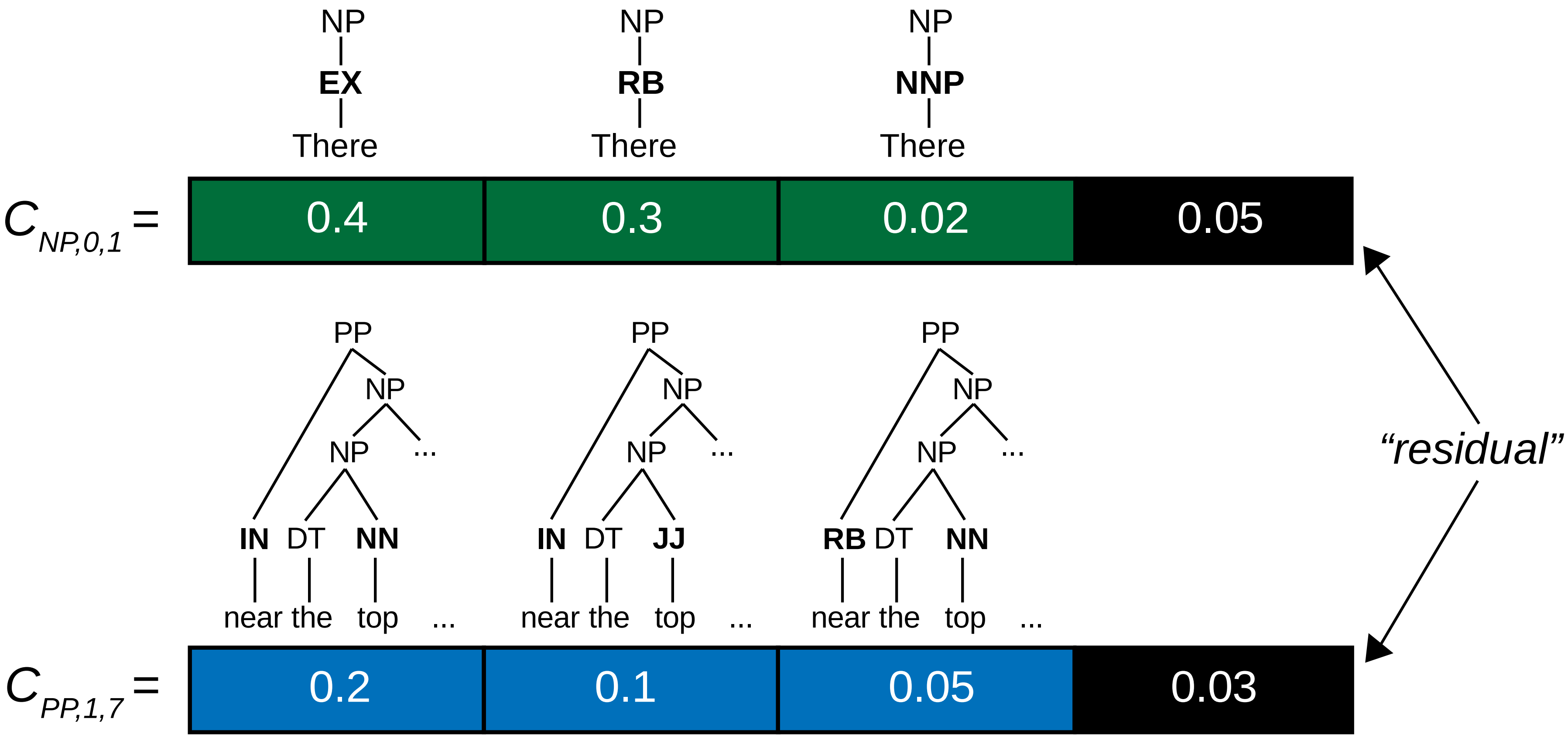
- Cube pruning does not actually expand all k^2 proofs as this example showed
- It uses an approximation that only looks at $O(k)$ proofs
- But since we are summing, we want to look at as many proofs as possible
- We use the algorithm that we just showed as the basis for cube summing (we call it **cube decoding** – details in paper)

Outline

- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion



ARK lti Carnegie Mellon

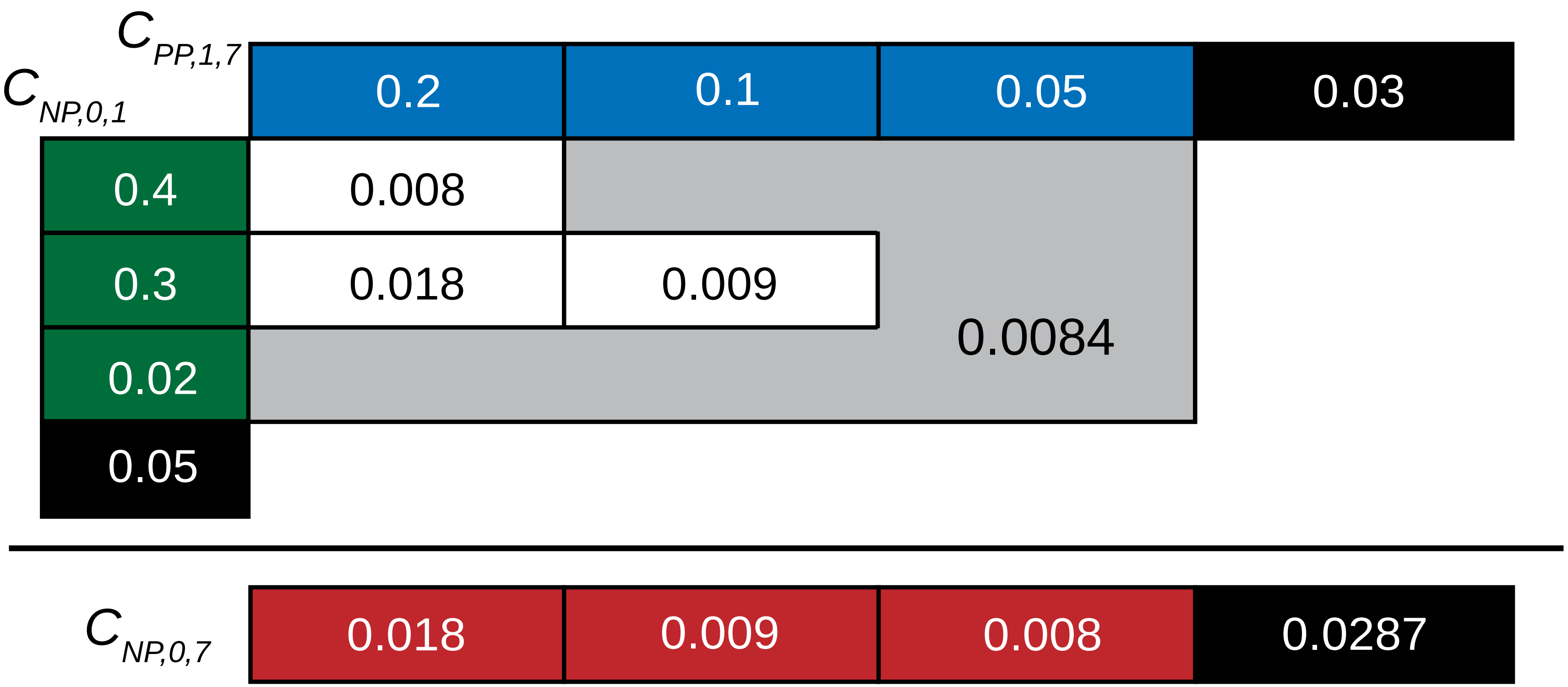

NP
EX
There
NP
RB
There
NP
NNP
There
C NP,0,1 =
0.4
0.3
0.02
0.05
PP
NP
NP
...
IN DT NN
near the top ...
PP
NP
NP
...
IN DT JJ
near the top ...
PP
NP
NP
...
RB DT NN
near the top ...
"residual"
C PP,1,7 =
0.2
0.1
0.05
0.03

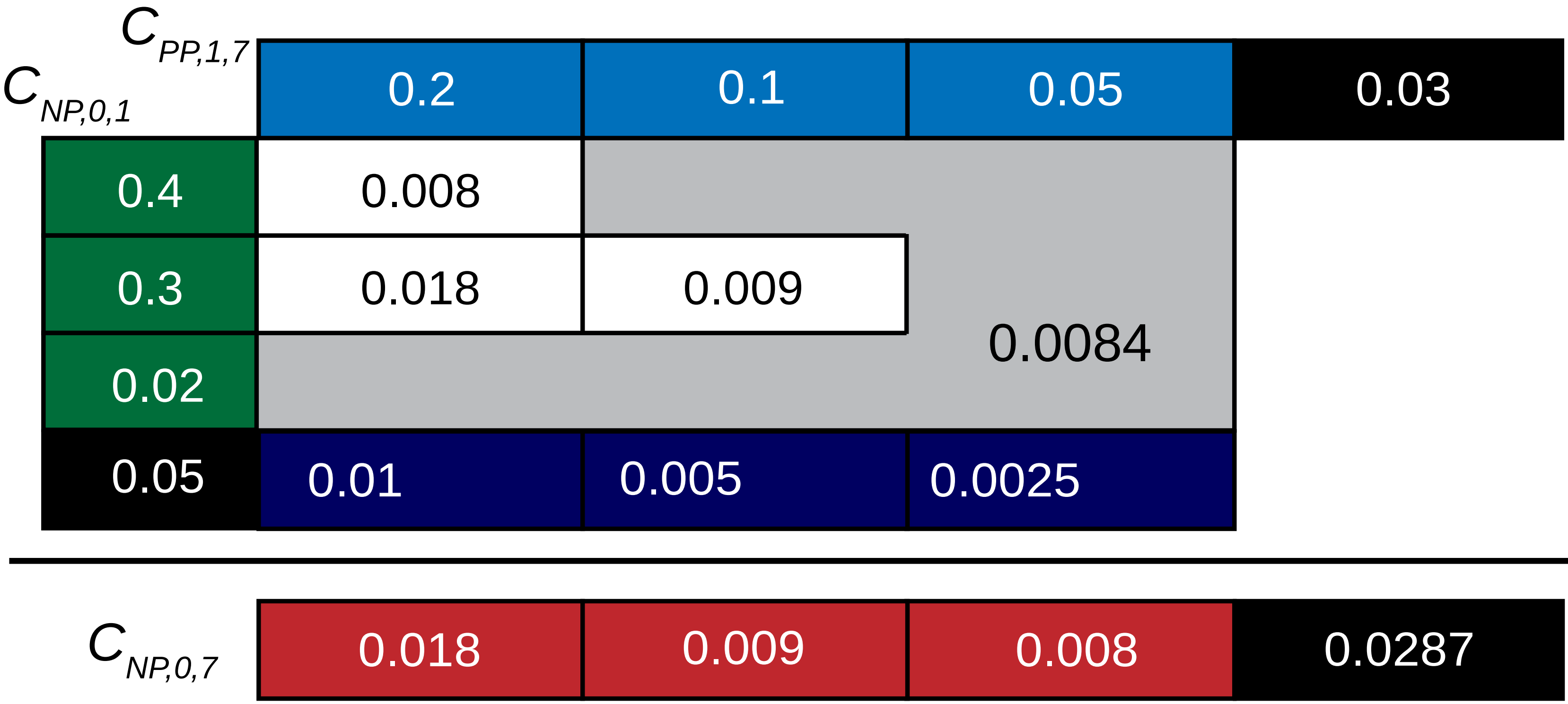
- Computation of local and non-local features is same as before
- Only difference is computing the residual for the result

$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05	0.03
0.4	0.008	0.004	0.001	
0.3	0.018	0.009	0.003	
0.02	0.0002	0.0001	0.0001	
0.05				

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287

ARK lti Carnegie Mellon


$C_{PP,1,7}$
$C_{NP,0,1}$
0.2
0.1
0.05
0.03
0.4
0.3
0.02
0.05
0.008
0.018
0.009
0.0084
$C_{NP,0,7}$
0.018
0.009
0.008
0.0287



$C_{PP,1,7}$
$C_{NP,0,1}$
0.2
0.1
0.05
0.03
0.4
0.008
0.3
0.018
0.009
0.0084
0.02
0.05
0.01
0.005
0.0025
$C_{NP,0,7}$
0.018
0.009
0.008
0.0287

$$C_{NP,0,7} = C_{NP,0,1} \times C_{PP,1,7} \times \lambda_{NP \rightarrow NP\ PP}$$

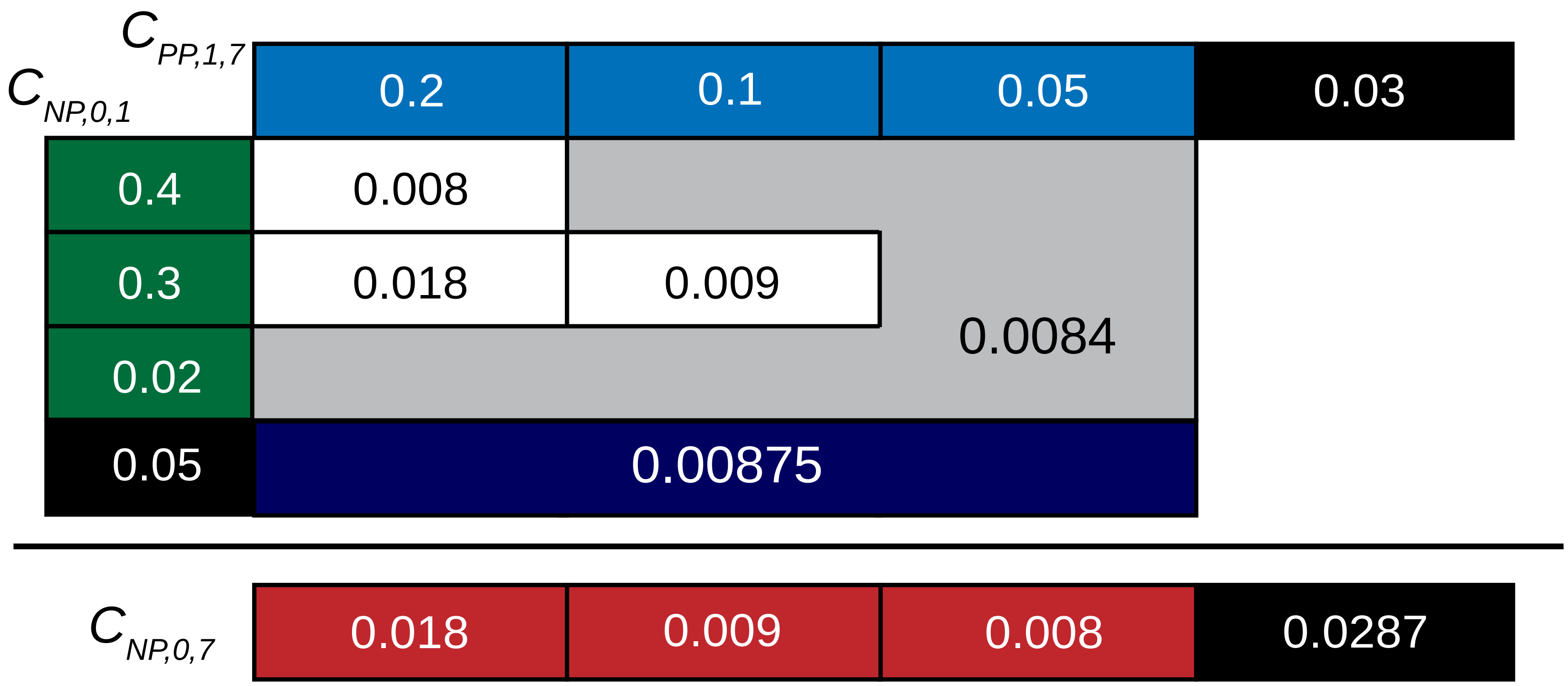
$$\lambda_{NP \rightarrow NP\ PP} = 0.5$$

$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05	0.03
0.4	0.008			
0.3	0.018	0.009	0.0084	
0.02				
0.05	0.01 × 0.5	0.005 × 0.5	0.0025 × 0.5	

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287

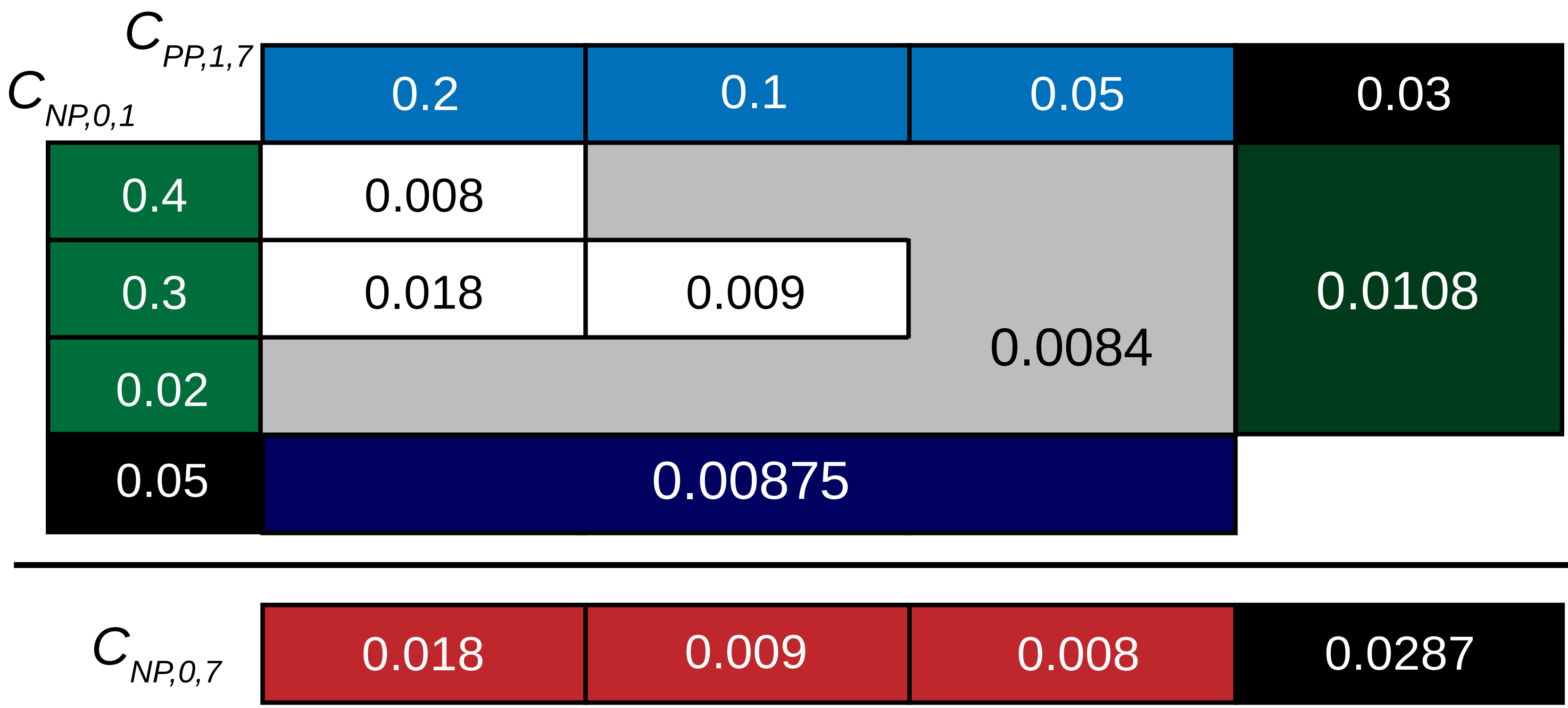
$C_{PP,1,7}$ / $C_{NP,0,1}$	0.2	0.1	0.05	0.03
0.4	0.008			
0.3	0.018	0.009		
0.02			0.0084	
0.05	0.005	0.0025	0.00125	

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287


$C_{PP,1,7}$
$C_{NP,0,1}$
0.2
0.1
0.05
0.03
0.4
0.008
0.3
0.018
0.009
0.02
0.0084
0.05
0.00875
$C_{NP,0,7}$
0.018
0.009
0.008
0.0287

$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05	0.03
0.4	0.008			0.012 × 0.5
0.3	0.018	0.009	0.0084	0.009 × 0.5
0.02				0.0006 × 0.5
0.05	0.00875			

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287



$C_{PP,1,7}$
$C_{NP,0,1}$
0.2
0.1
0.05
0.03
0.4
0.008
0.3
0.018
0.009
0.0084
0.0108
0.02
0.05
0.00875
$C_{NP,0,7}$
0.018
0.009
0.008
0.0287

$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05	0.03
0.4	0.008			0.0108
0.3	0.018	0.009	0.0084	
0.02				
0.05	0.00875			0.0015 × 0.5

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287

<table>
<tr><td>$C_{NP,0,1}$ \ $C_{PP,1,7}$</td><td>0.2</td><td>0.1</td><td>0.05</td><td>0.03</td></tr>
<tr><td>0.4</td><td>0.008</td><td colspan="2"></td><td rowspan="3">0.0108</td></tr>
<tr><td>0.3</td><td>0.018</td><td>0.009</td><td rowspan="2">0.0084</td></tr>
<tr><td>0.02</td><td colspan="2"></td></tr>
<tr><td>0.05</td><td colspan="3">0.00875</td><td>0.00075</td></tr>
</table>

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287

$$0.0287 = 0.0084 + 0.00875 + 0.0108 + 0.00075$$

$C_{NP,0,1}$ \ $C_{PP,1,7}$	0.2	0.1	0.05	0.03
0.4	0.008			0.0108
0.3	0.018	0.009	0.0084	
0.02				
0.05	0.00875			0.00075

$C_{NP,0,7}$	0.018	0.009	0.008	0.0287

Summary

- Maintain residual sum of all proofs not in k-best list
- Redefine operations to update the residual as necessary
- Result is approximate k-best proof list for $goal$ and approximate sum of all other proofs of $goal$
- When $k = \infty$, result is exact

Outline

- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion

Semirings

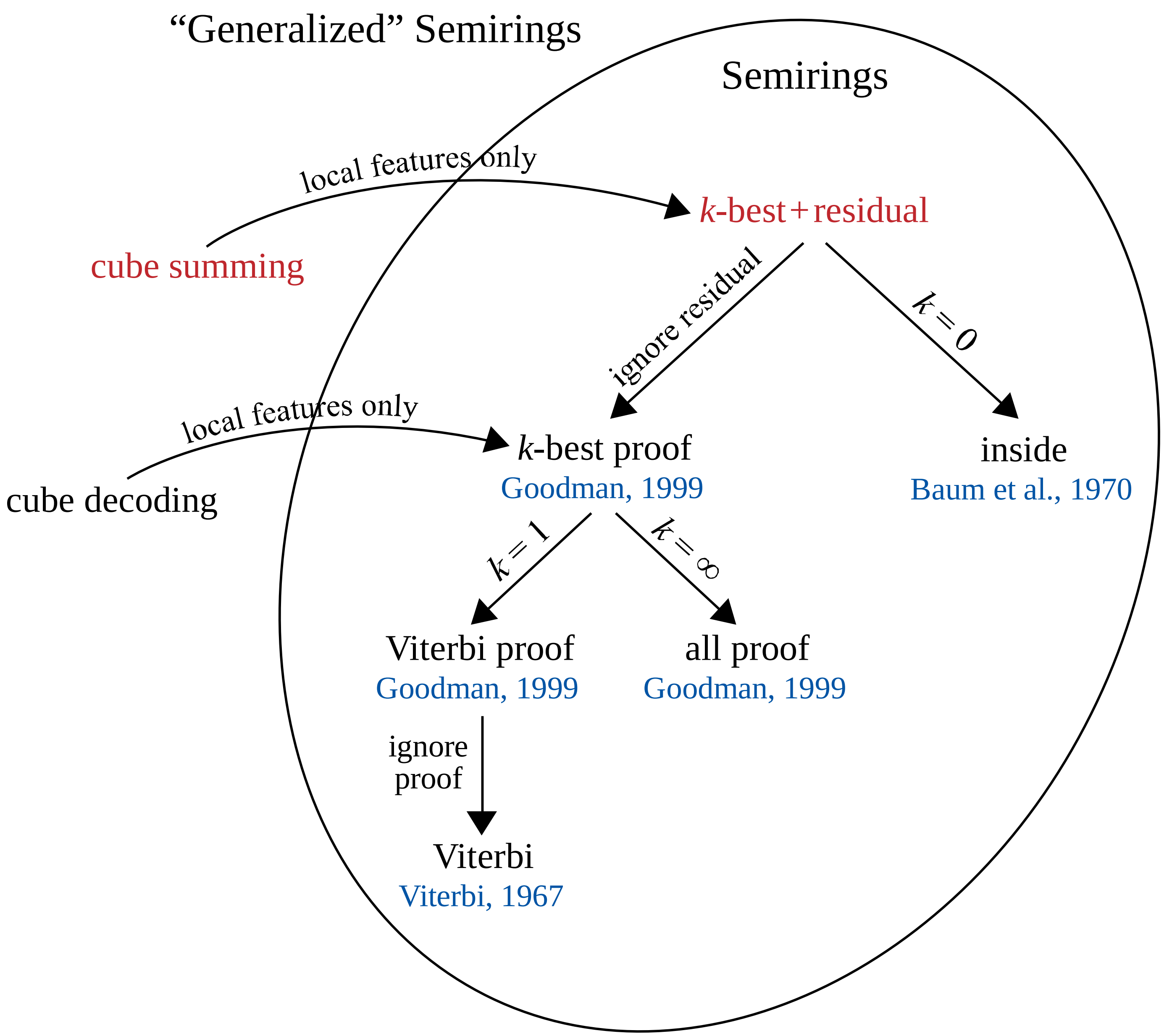
Semiring	A	$\oplus$	$\otimes$	$\mathbf{0}$	$\mathbf{1}$
Inside	$\mathbb{R}_{\geq 0}$	$a+b$	ab	0	1
Viterbi	$\mathbb{R}_{\geq 0}$	$\max(a,b)$	ab	0	1

- A *semiring* is a tuple $\langle A, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ such that:
  - $\oplus : A \times A \rightarrow A$ is associative and commutative
  - $\otimes : A \times A \rightarrow A$ is associative and distributes over $\oplus$
  - $\forall a \in A, a \oplus \mathbf{0} = a,$
    $a \otimes \mathbf{1} = a,$
    $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$

ARK lti Carnegie Mellon

Non-local features break some of the semiring properties!

(see paper for details)


"Generalized" Semirings
Semirings
local features only
k-best + residual
cube summing
ignore residual
$k = 0$
local features only
k-best proof
Goodman, 1999
inside
Baum et al., 1970
cube decoding
$k = 1$
$k = \infty$
Viterbi proof
Goodman, 1999
all proof
Goodman, 1999
ignore proof
Viterbi
Viterbi, 1967

Outline

- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion

Implementation

- Several implementation tools exist for dynamic programming
  - Dyna (Eisner et al., 2005) and Goodman (1999) assume semirings
  - Hypergraphs (Klein & Manning, 2001; Huang, 2008) do not require semirings but are aimed at decoding

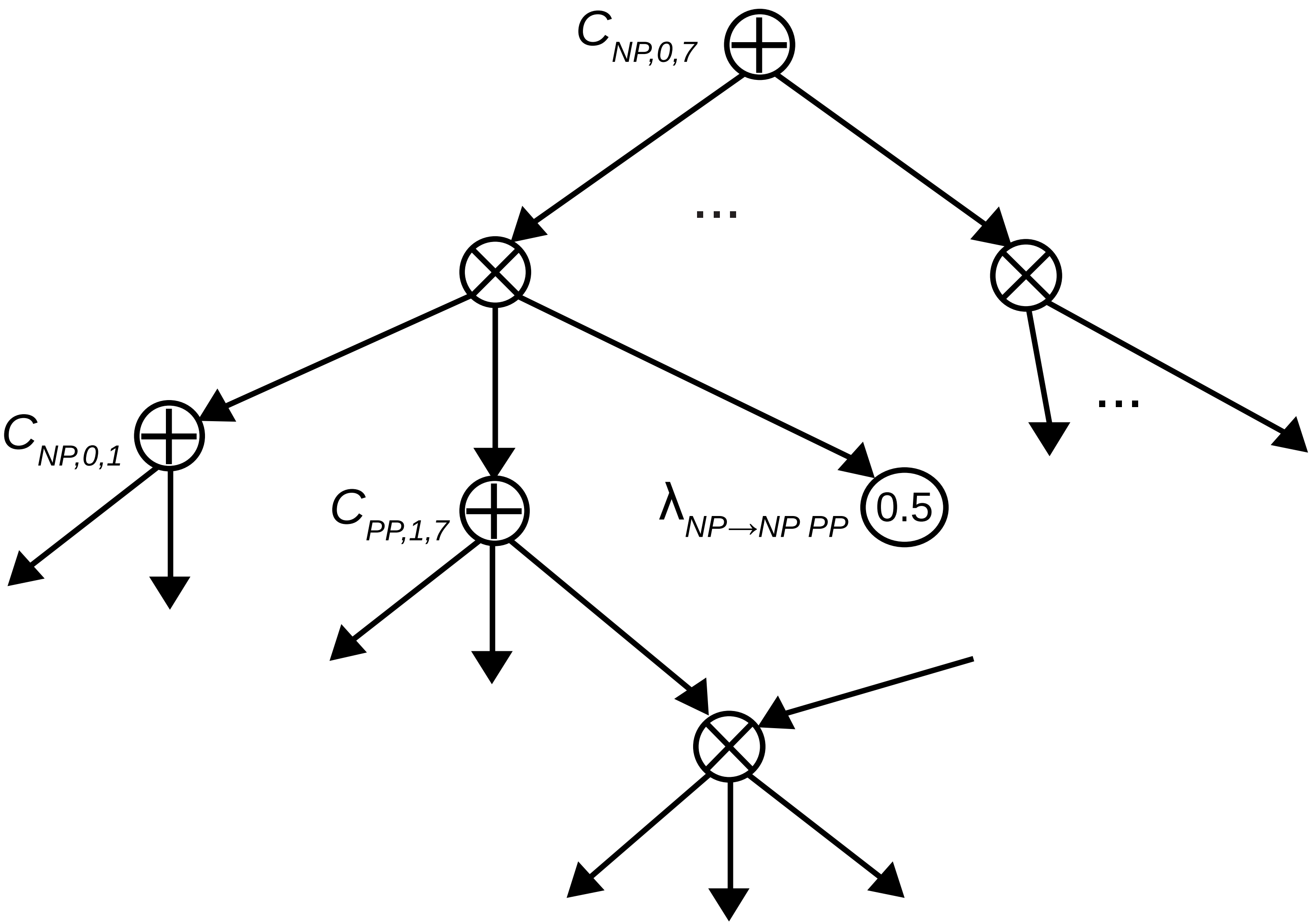
- These could be extended for cube summing, but we instead use a lower-level formalism: **arithmetic circuits**

Arithmetic Circuits

- Explicitly represent computations to be performed using a directed graph
  - Operators and operands are nodes in the graph
  - A value is associated with each node
  - Operators point to their operands

- Allow automatic differentiation in the reverse mode (Griewank & Corliss, 1991) for efficient gradient computation

Example



Outline

- Background
- Cube Pruning
- Cube Summing
- Semirings
- Implementation
- Conclusion

Conclusion and Ongoing Work

- We have described cube summing, a technique for approximate summing using dynamic programming with non-local features
- With only local features, cube summing is a semiring that generalizes those in common use
- Some semiring properties are broken by non-local features but an implementation based on arithmetic circuits can be used
- We are currently using cube summing to train a log-linear syntactic translation model with hidden variables

ARK lti Carnegie Mellon

Thanks!

Cube Summing, Approximate Inference with Non-Local Features, and Dynamic Programming without Semirings

Kevin Gimpel and Noah A. Smith