Integrated CCG Parsing and Supertagging

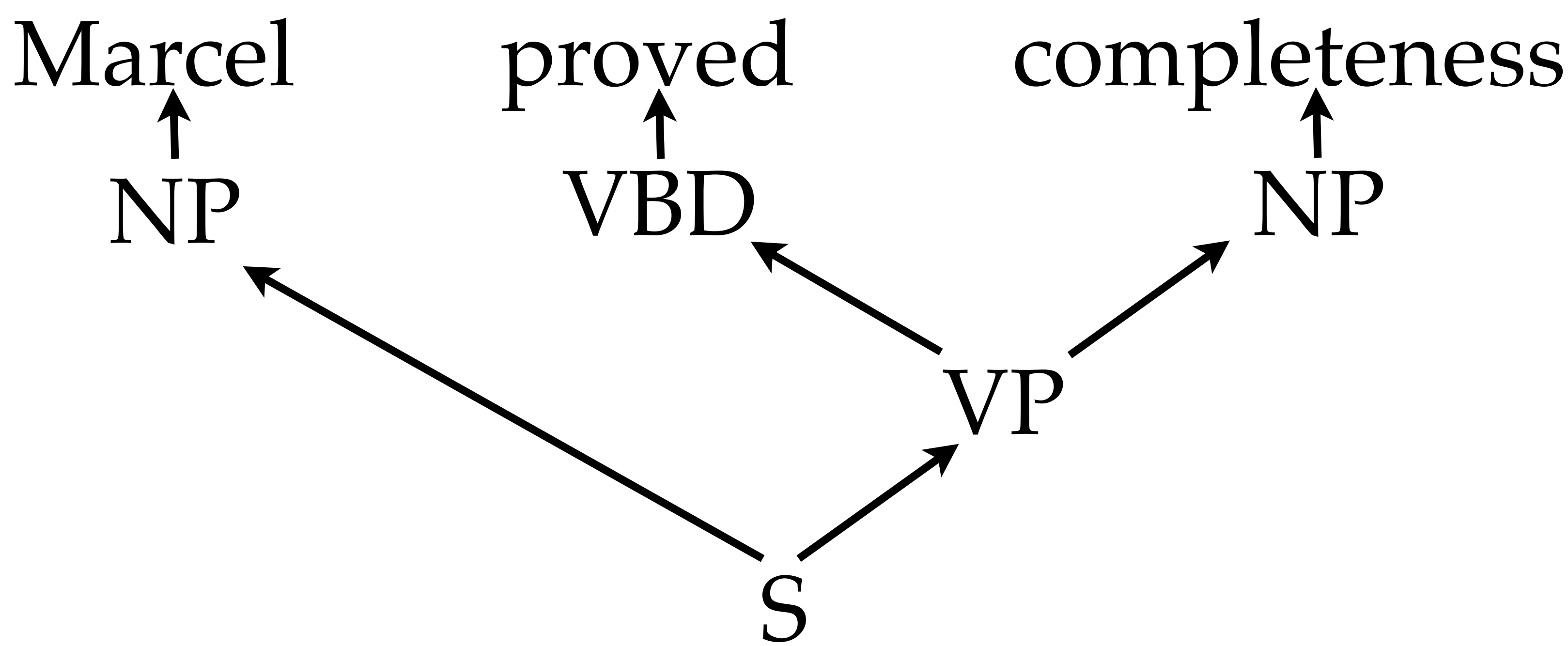
Michael Auli
(University of Edinburgh; graduating spring 2012)
joint work with
Adam Lopez (Johns Hopkins University)

Parsing

Parsing

Marcel proved completeness

Parsing



Parsing

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

Marcel proved completeness

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

$$\frac{\text{Marcel}}{NP} \qquad \frac{\text{proved}}{(S\backslash NP)/NP} \qquad \frac{\text{completeness}}{NP}$$

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

$$
\dfrac{\text{Marcel}}{NP} \quad \dfrac{\text{proved}}{(S\backslash NP)/\color{red}{NP}} \quad \dfrac{\text{completeness}}{\color{red}{NP}}
$$

$$
\dfrac{(S\backslash NP)/\color{red}{NP} \quad \color{red}{NP}}{\mathrm{S\backslash NP}}>
$$

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

$$\begin{array}{ccc} \text{Marcel} & \text{proved} & \text{completeness} \\ \hline NP & (S\backslash NP)/{\color{red}NP} & {\color{red}NP} \\ \cline{2-3} & \multicolumn{2}{c}{\mathrm{S}\backslash\mathrm{NP} \quad >} \end{array}$$

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

$$
\dfrac{\dfrac{\text{Marcel}}{NP} \qquad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP} \qquad \dfrac{\text{completeness}}{NP}}{S\backslash NP}\,>}{S}\,<
$$

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

Marcel proved completeness

$$\dfrac{\dfrac{\text{Marcel}}{NP} \quad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP} \quad \dfrac{\text{completeness}}{NP}}{S\backslash NP}>}{S}<$$

Combinatory Categorial Grammar (CCG; Steedman 2000)

lexicalized, binary branching, CKY-parseable

Parsing

time flies like an arrow

Parsing

$$
\begin{array}{ccccc}
\underline{\text{time}} & \underline{\text{flies}} & \underline{\text{like}} & \underline{\text{an}} & \underline{\text{arrow}} \\
NP & S\backslash NP & ((S\backslash NP)\backslash(S\backslash NP))/NP & NP/NP & NP
\end{array}
$$

Parsing

time	flies	like	an	arrow
NP	$S\backslash NP$	$((S\backslash NP)\backslash(S\backslash NP))/NP$	NP/NP	NP
N/N	NP	PP/NP	$(NP\backslash NP)/N$	$NP\backslash NP$
N/S	$(S\backslash NP)/PP$	$(NP\backslash NP)/NP$	$((S\backslash NP)\backslash(S\backslash NP))/N$	N/N
$(S\backslash NP)\backslash(S\backslash NP)$	$(S\backslash NP)/S$	$(S/S)/NP$	$(S\backslash S)/N$	N/S
$NP\backslash NP$	$(S\backslash NP)/NP$	$((S\backslash NP)/(S\backslash NP))/S$	NP	$(\mathrm{S}\backslash\mathrm{NP})\backslash(\mathrm{S}\backslash\mathrm{NP})$
$(S\backslash NP)/NP$	$((S\backslash NP)/PP)/NP$	$(PP/PP)/NP$	$(N/N)/(N/N)$	$\mathrm{S}\backslash\mathrm{NP}$
...	$(S\backslash NP)/S$	PP/S	...	...
	...	...		

Parsing

time	flies	like	an	arrow
NP	$S\backslash NP$	$((S\backslash NP)\backslash(S\backslash NP))/NP$	NP/NP	NP
N/N	NP	PP/NP	$(NP\backslash NP)/N$	$NP\backslash NP$
N/S	$(S\backslash NP)/PP$	$(NP\backslash NP)/NP$	$((S\backslash NP)\backslash(S\backslash NP))/N$	N/N
$(S\backslash NP)\backslash(S\backslash NP)$	$(S\backslash NP)/S$	$(S/S)/NP$	$(S\backslash S)/N$	N/S
$NP\backslash NP$	$(S\backslash NP)/NP$	$((S\backslash NP)/(S\backslash NP))/S$	NP	(S\NP)\(S\NP)
$(S\backslash NP)/NP$	$((S\backslash NP)/PP)/NP$	$(PP/PP)/NP$	$(N/N)/(N/N)$	S\NP
...	$(S\backslash NP)/S$	PP/S	...	...
	...	...		

Over 22 tags per word! (Clark & Curran 2004)

Hard parsing task

Supertagging

Supertagging

time flies like an arrow

Supertagging

$$\frac{\text{time}}{NP} \quad \frac{\text{flies}}{S\backslash NP} \quad \frac{\text{like}}{((S\backslash NP)\backslash(S\backslash NP))/NP} \quad \frac{\text{an}}{NP/NP} \quad \frac{\text{arrow}}{NP}$$

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$((S\backslash NP)\backslash(S\backslash NP))/NP$	NP/NP	NP

$$\text{an arrow: } NP/NP \;\; NP \xrightarrow{>} \mathrm{NP}$$

$$\text{like an arrow: } ((S\backslash NP)\backslash(S\backslash NP))/NP \;\; \mathrm{NP} \xrightarrow{>} \mathrm{(S\backslash NP)\backslash(S\backslash NP)}$$

$$\text{flies like an arrow: } S\backslash NP \;\; \mathrm{(S\backslash NP)\backslash(S\backslash NP)} \xrightarrow{<} \mathrm{S\backslash NP}$$

$$\text{time flies like an arrow: } NP \;\; \mathrm{S\backslash NP} \xrightarrow{<} \mathrm{S}$$

Overview

- The Problem with Pipeline Models
- Decoding Integrated Models
  with Loopy Belief Propagation and Dual Decomposition (ACL 2011)
- Training Integrated Models
  with Softmax-Margin using Exact and Approximate Loss Functions (EMNLP 2011)

Overview

- The Problem with Pipeline Models
- Decoding Integrated Models
  with Loopy Belief Propagation and Dual Decomposition (ACL 2011)
- Training Integrated Models
  with Softmax-Margin using Exact and Approximate Loss Functions (EMNLP 2011)

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP
			$\xrightarrow{\quad\quad} >$	
			NP	
		$\xrightarrow{\quad\quad} >$		
		S\NP		

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP

$$\dfrac{NP/NP \quad NP}{\mathrm{NP}} >$$

$$\dfrac{(S\backslash NP)/NP \quad \mathrm{NP}}{\mathrm{S\backslash NP}} >$$

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP

Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP
NP/NP	NP	....	...	...
...	...	$((S\backslash NP)\backslash(S\backslash NP))/NP$		
		....		

Adaptive Supertagging

Clark & Curran (2004)

Adaptive Supertagging

Clark & Curran (2004)

- Algorithm:
  - Run supertagger.
  - Return tags with posterior higher than some alpha.
  - Parse by combining tags (CKY).
  - If parsing succeeds, stop.
  - If parsing fails, lower alpha and repeat.

Adaptive Supertagging

Clark & Curran (2004)

- Algorithm:
  - Run supertagger.
  - Return tags with posterior higher than some alpha.
  - Parse by combining tags (CKY).
  - If parsing succeeds, stop.
  - If parsing fails, lower alpha and repeat.
- Q: are parses returned in early rounds suboptimal?

Answer...

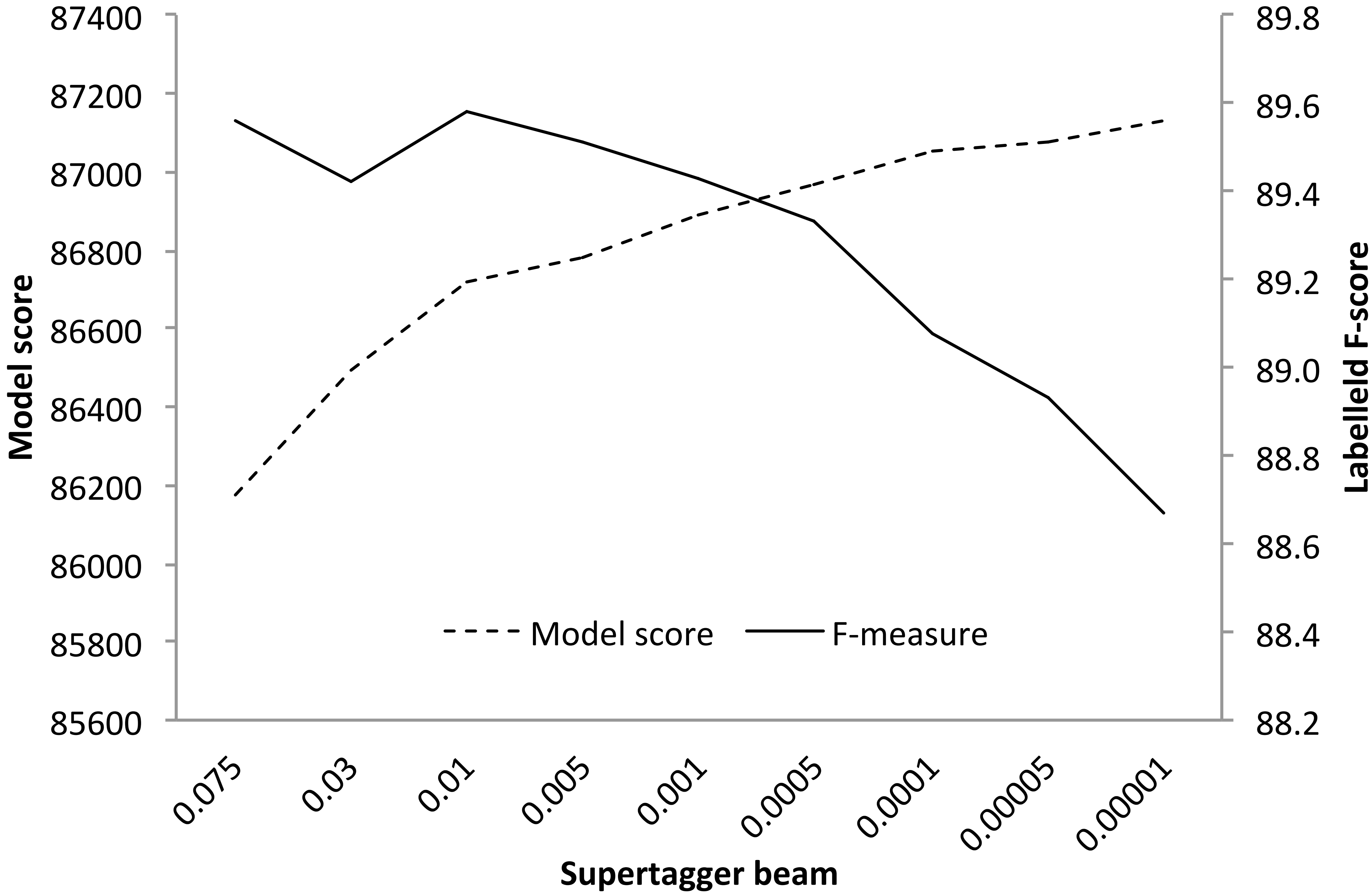
Answer...

- Oracle parsing (Huang 2008):
  - With tight beam: 94.35
  - With loose beam: **97.65**
  - With gold-supertags: 97.73

Answer...

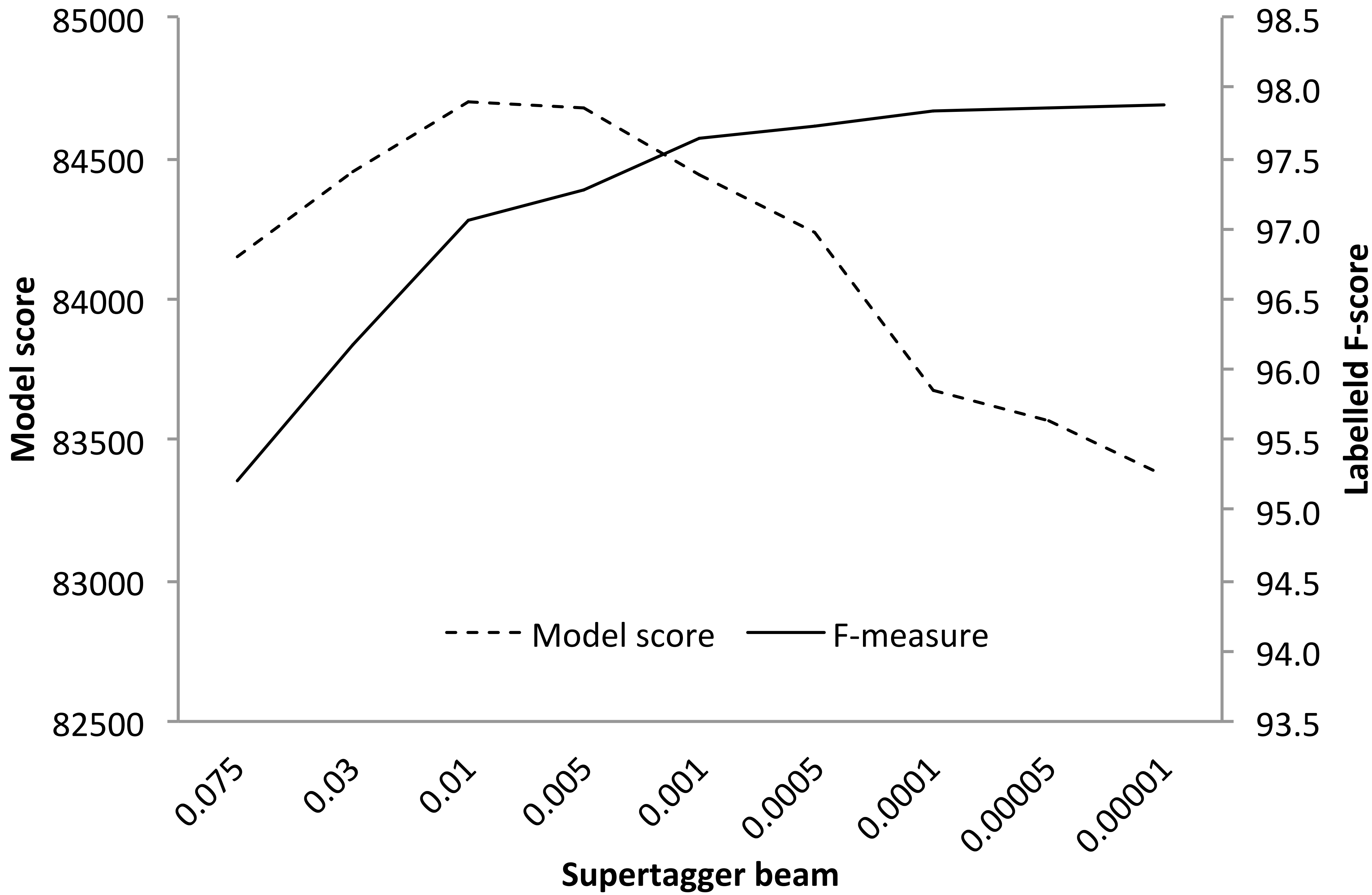
- Oracle parsing (Huang 2008):
  - With tight beam: 94.35
  - With loose beam: **97.65**
  - With gold-supertags: 97.73
- Standard parsing task (Clark & Curran 2007):
  - With tight beam: 87.38 (labeled F-measure)
  - With loose beam: 87.36

Parsing



Note: only sentences parsable at all beam settings.

Oracle Parsing



Note: only sentences parsable at all beam settings.

What’s happening here?

What's happening here?

- Supertagger keeps parser from making serious errors.
- But it also occasionally prunes away useful parses.

What’s happening here?

- Supertagger keeps parser from making serious errors.
- But it also occasionally prunes away useful parses.
- Why not combine supertagger and parser into one?

Overview

- The Problem with Pipeline Models
- Decoding Integrated Models
  with Loopy Belief Propagation and Dual Decomposition (ACL 2011)
- Training Integrated Models
  with Softmax-Margin using Exact and Approximate Loss Functions (EMNLP 2011)

Integrated Model

- Supertagger & parser are undirected log-linear models.
- **Idea**: combine their features into one model.
- **Problem**: Exact computation of marginal or maximum quantities becomes very expensive because parsing and tagging submodels must agree on the tag sequence.

Integrated Model

- Supertagger & parser are undirected log-linear models.
- **Idea**: combine their features into one model.
- **Problem**: Exact computation of marginal or maximum quantities becomes very expensive because parsing and tagging submodels must agree on the tag sequence.

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Integrated Model

$\mathrm{A} \rightarrow \mathrm{B} \;\; \mathrm{C} \qquad O(Gn^3)$

$$
\begin{array}{ccc}
\underline{\text{Marcel}} & \underline{\text{proved}} & \underline{\text{completeness}} \\
NP & (S\backslash NP)/NP & NP \\
\cline{2-3}
 & \multicolumn{2}{c}{\mathrm{S}\backslash\mathrm{NP}} \quad > \\
\hline
\multicolumn{3}{c}{\mathrm{S}} \quad <
\end{array}
$$

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Integrated Model

$$A \rightarrow B \quad C \qquad O(Gn^3)$$

$${}_{q}A_{r} \rightarrow {}_{q}B_{s} \ {}_{s}C_{r} \qquad O(G^3n^3)$$

$$\begin{array}{ccc} \underline{\text{Marcel}} & \underline{\text{proved}} & \underline{\text{completeness}} \\ {}_{<s>}NP_{(S\backslash NP)/NP} & {}_{NP}(S\backslash NP)/NP_{NP} & {}_{(S\backslash NP)/NP}\,NP_{</s>} \end{array}$$

$$\underline{\qquad\qquad {}_{NP}\,S\backslash NP_{</s>} \qquad\qquad} >$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad} <$$

$${}_{<s>}\,S_{</s>}$$

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Approximate Algorithms

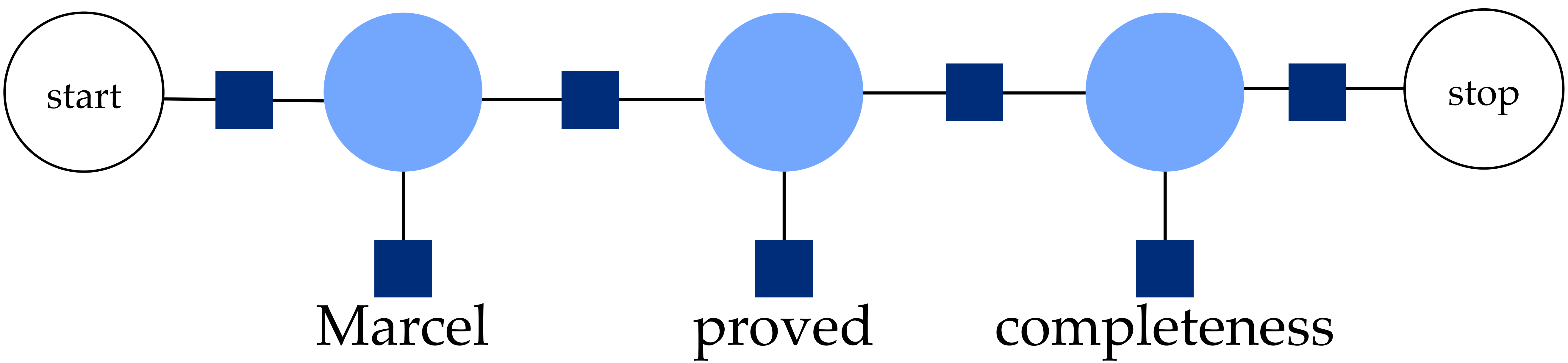
- Loopy belief propagation: approximate calculation of marginals. (Pearl 1988; Smith & Eisner 2008)
- Dual decomposition: exact (sometimes) calculation of maximum. (Dantzig & Wolfe 1960; Komodakis et al. 2007; Koo et al. 2010)

Belief Propagation

Belief Propagation

Forward-backward is belief propagation (Smyth et al. 1997)

Belief Propagation



Forward-backward is belief propagation (Smyth et al. 1997)

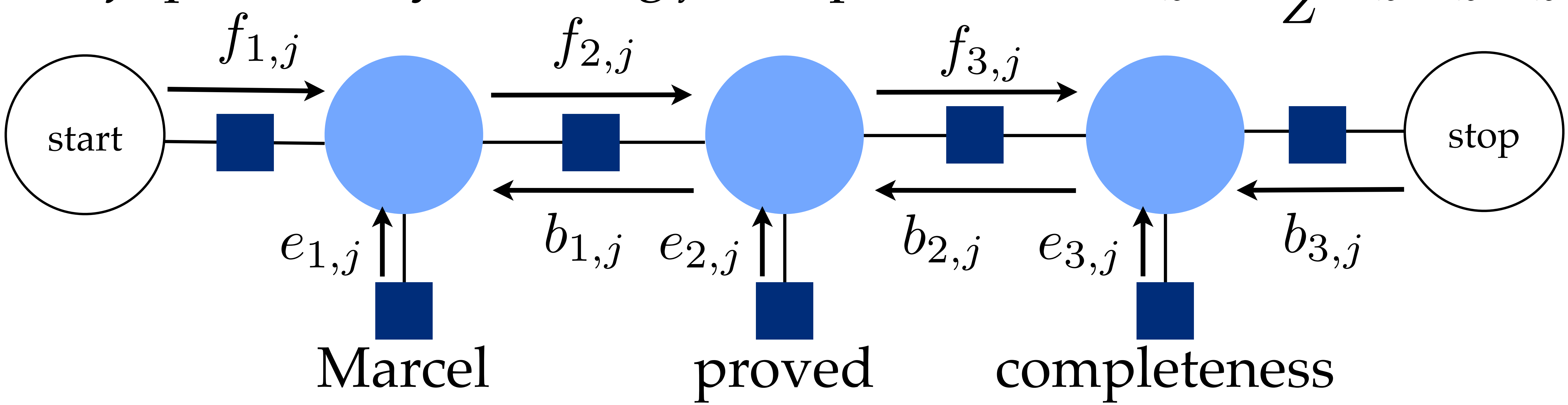
Belief Propagation

emission message: $e_{i,j}$

forward message: $f_{i,j} = \sum_{j'} f_{i-1,j'} e_{i-1,j'} t_{j',j}$

backward message: $b_{i,j} = \sum_{j'} b_{i+1,j'} e_{i+1,j'} t_{j,j'}$

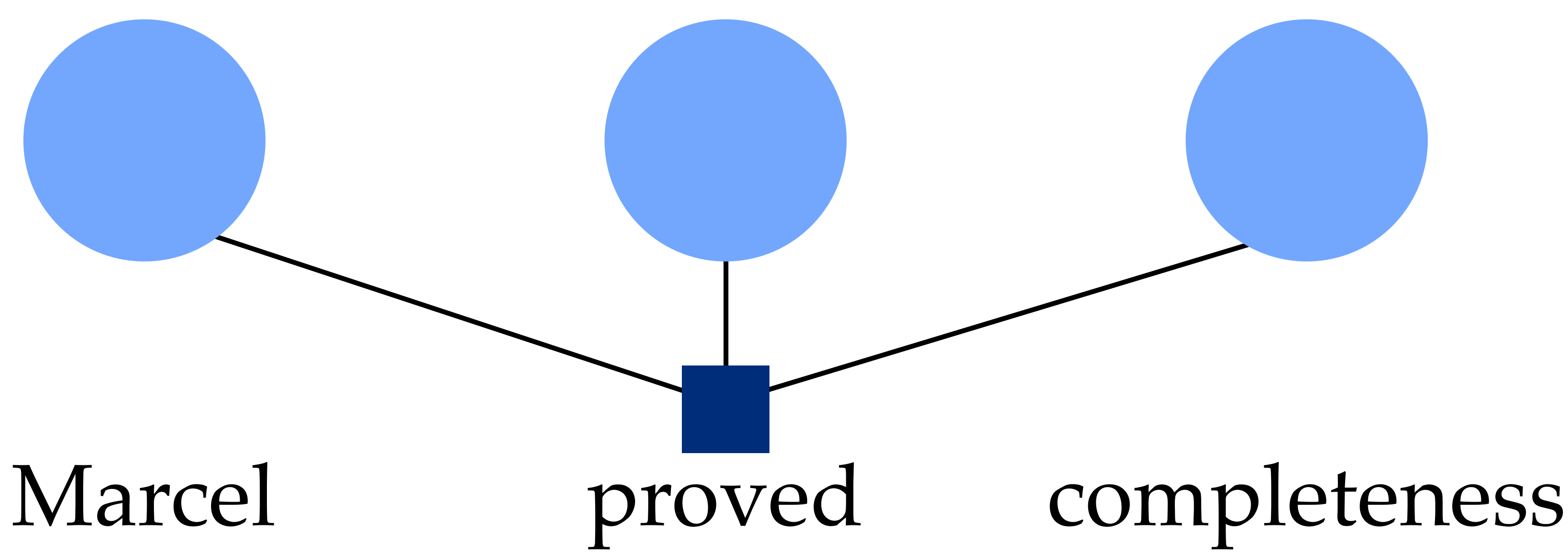
belief (probability) that tag *j* is at position *i*: $p_{i,j} = \frac{1}{Z} f_{i,j} e_{i,j} b_{i,j}$



Forward-backward is belief propagation (Smyth et al. 1997)

Belief Propagation

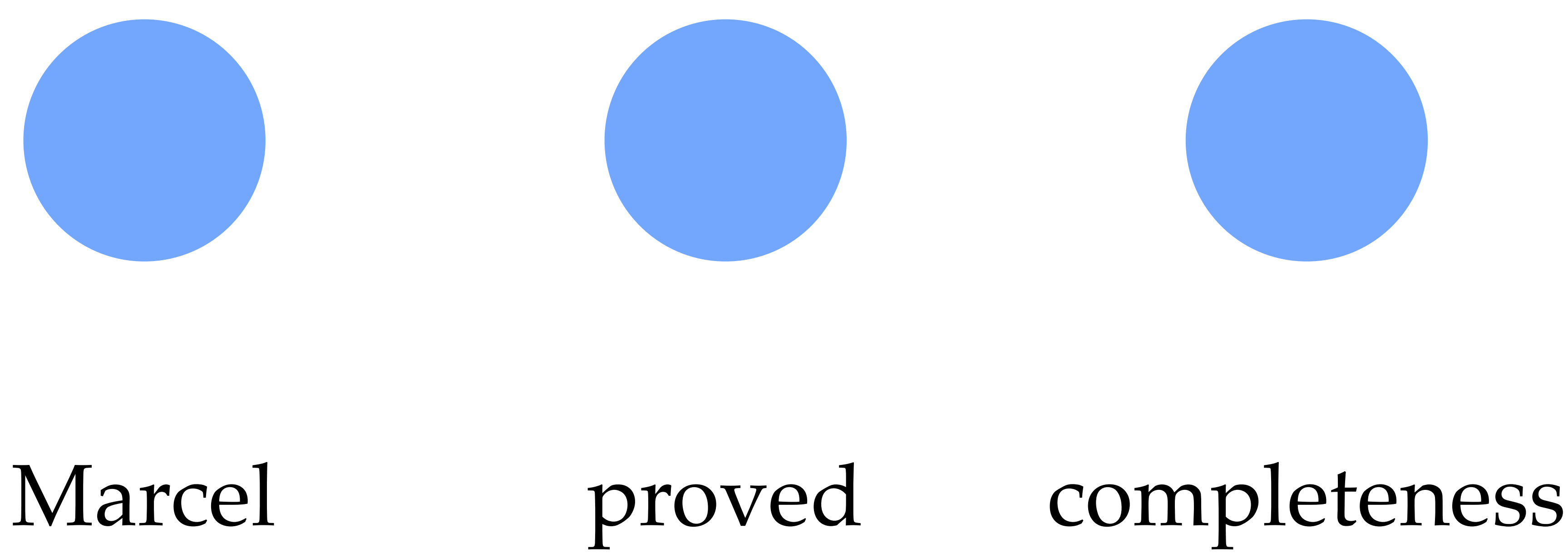
Notational convenience: one factor describes whole distribution over supertag sequence...



Belief Propagation

We can also do the same for the distribution over parse trees

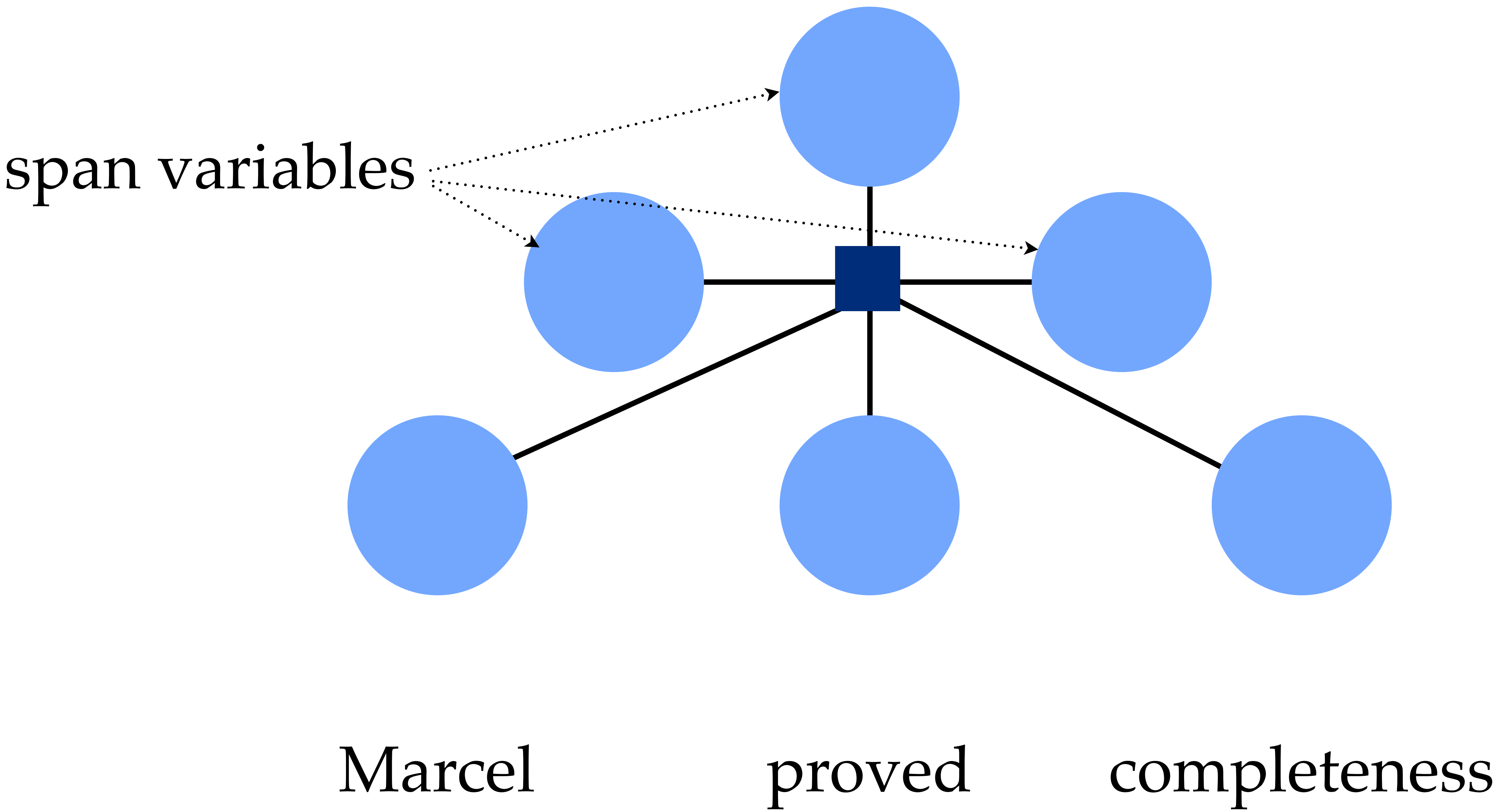
(Case-factor diagrams: McAllester et al. 2008)



Belief Propagation

We can also do the same for the distribution over parse trees

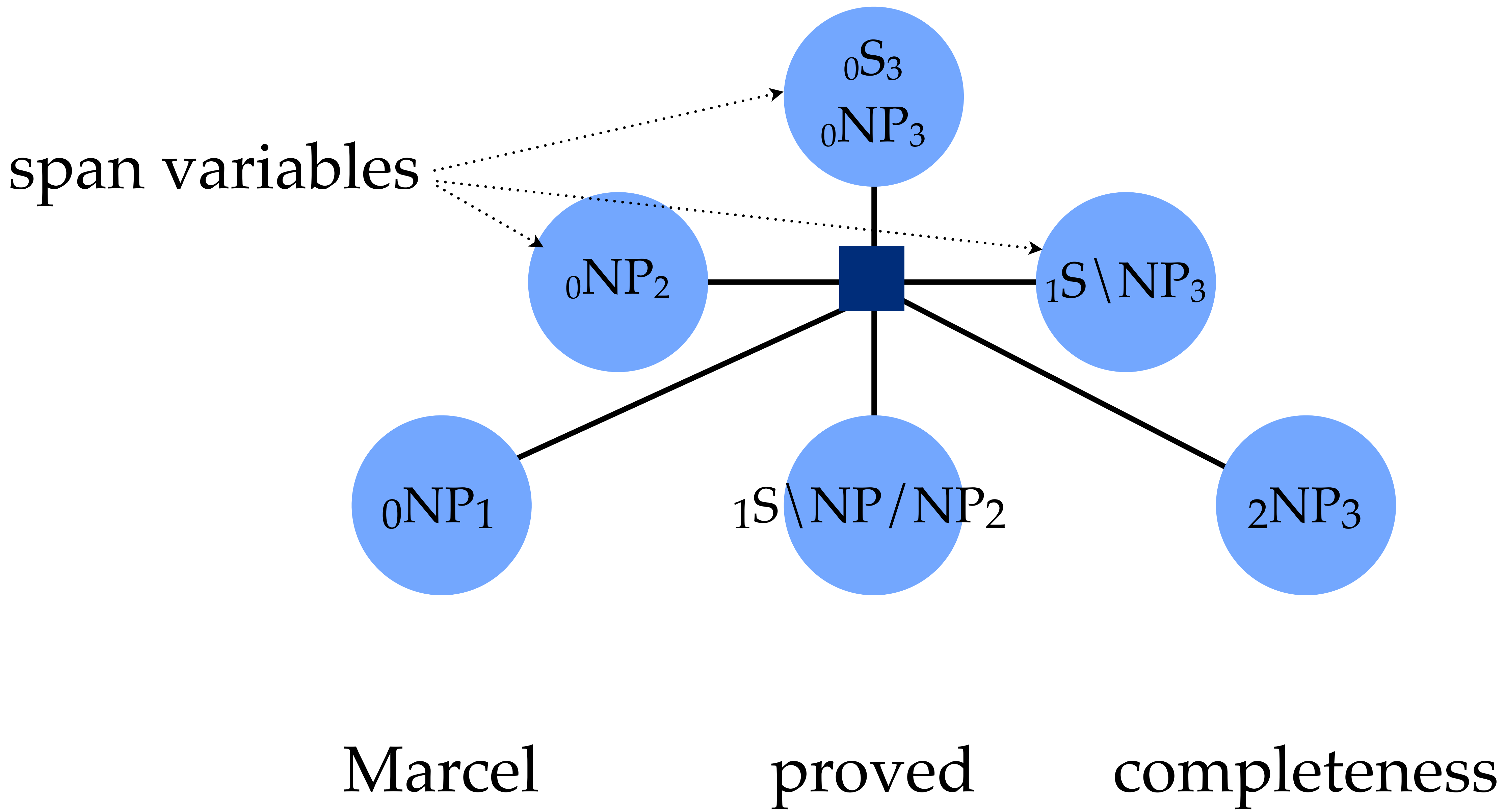
(Case-factor diagrams: McAllester et al. 2008)



Belief Propagation

We can also do the same for the distribution over parse trees

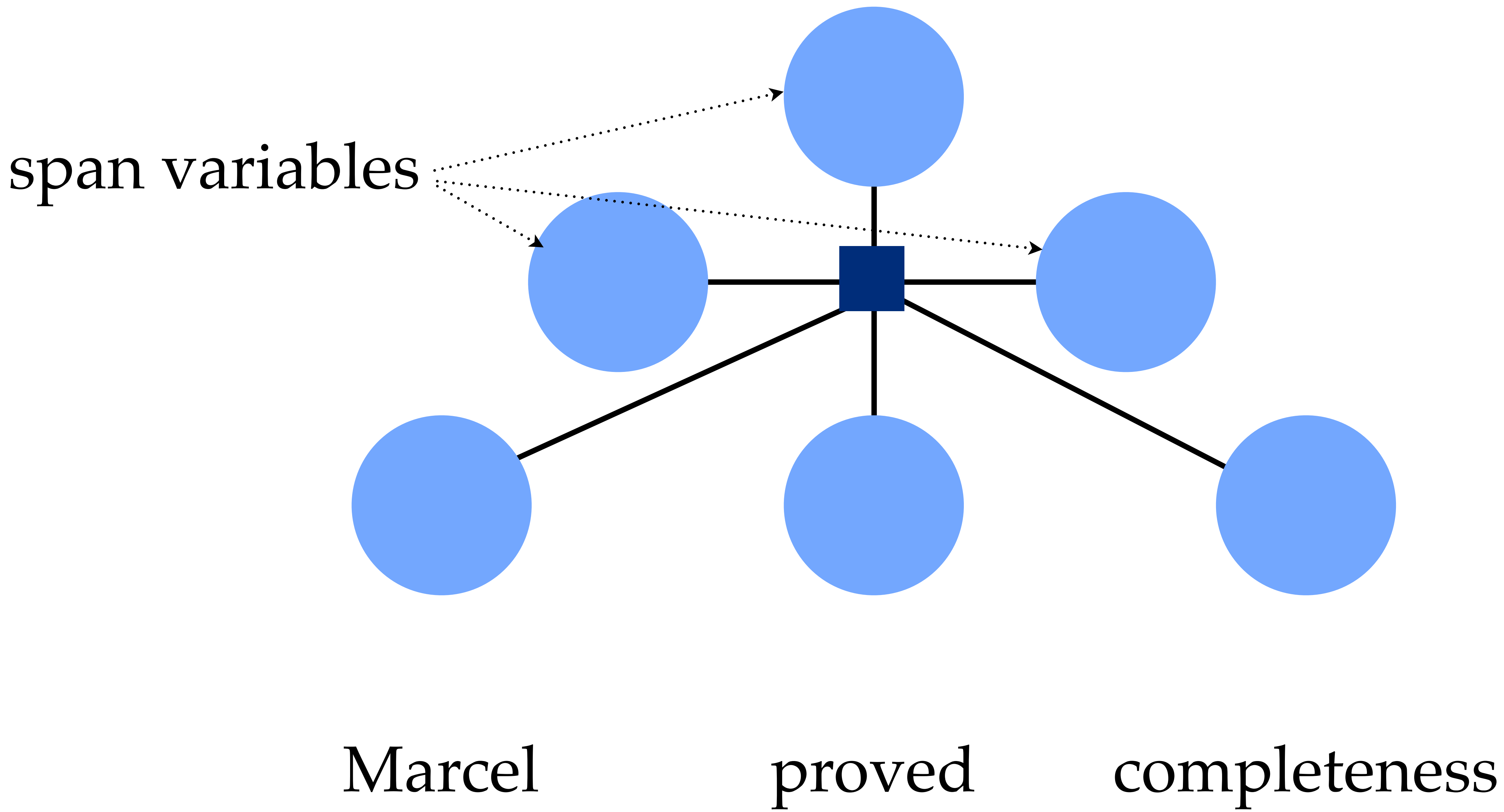
(Case-factor diagrams: McAllester et al. 2008)



Belief Propagation

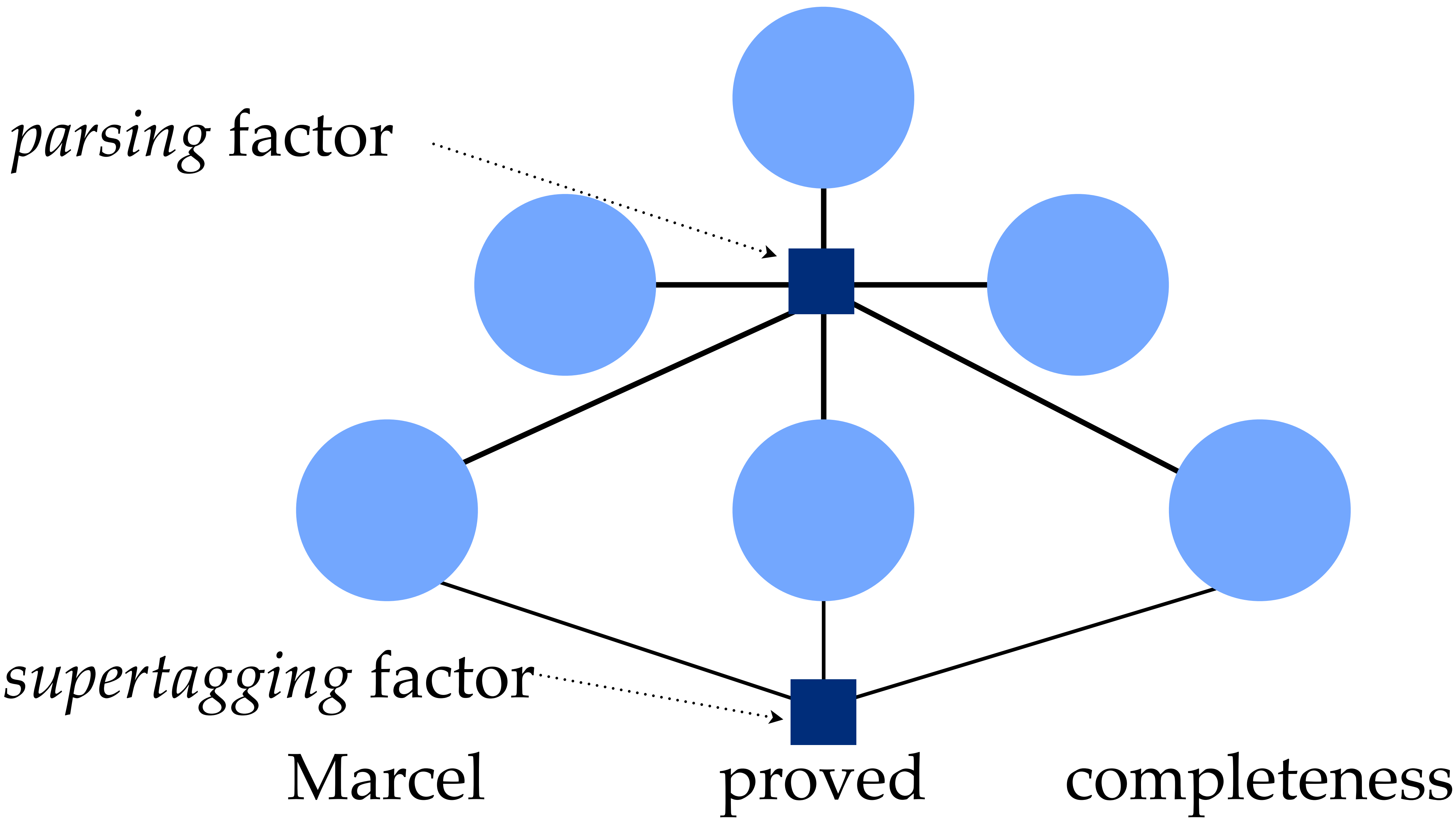
We can also do the same for the distribution over parse trees

(Case-factor diagrams: McAllester et al. 2008)



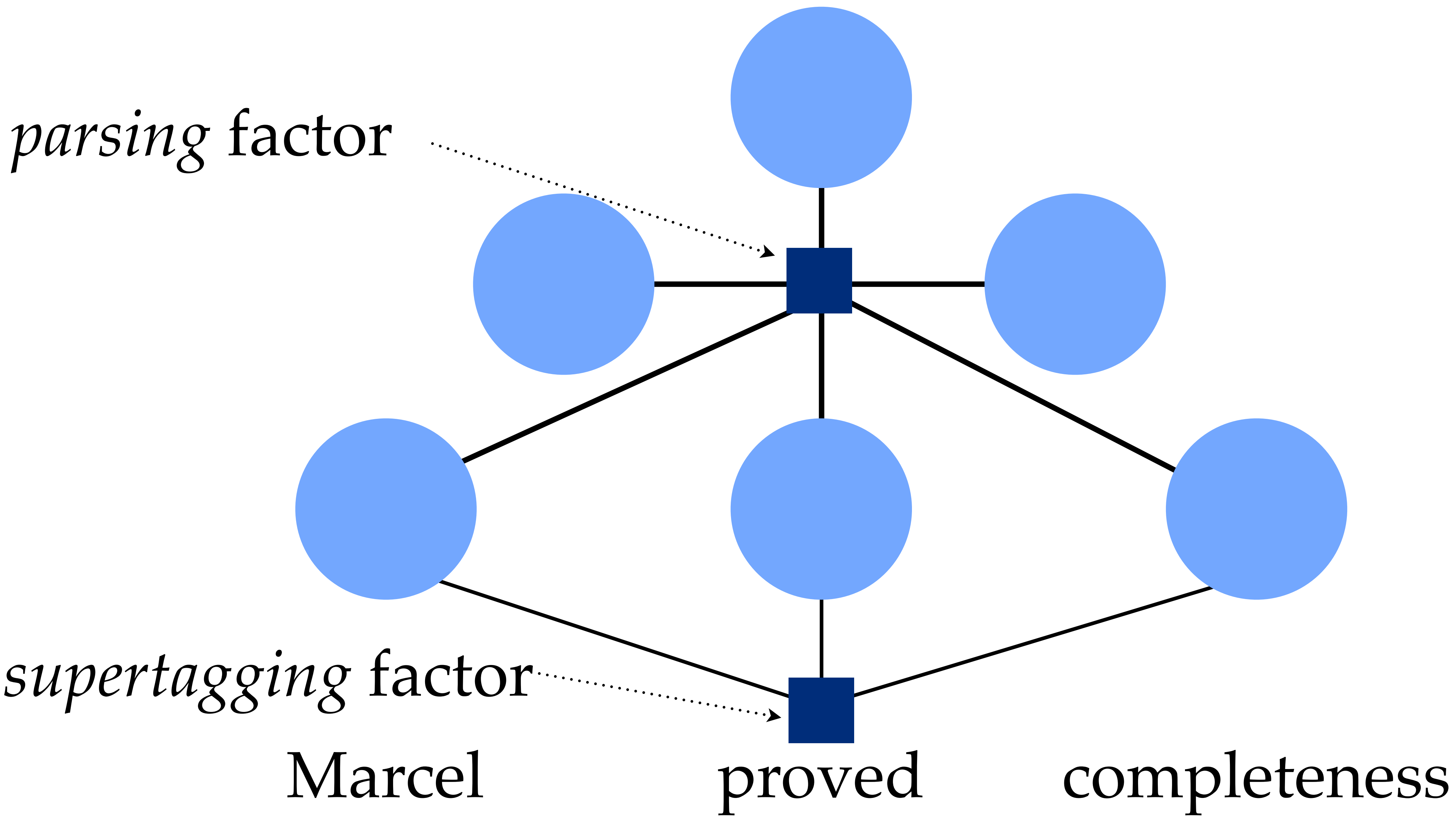
Inside-outside is belief propagation (Sato 2007)

Belief Propagation



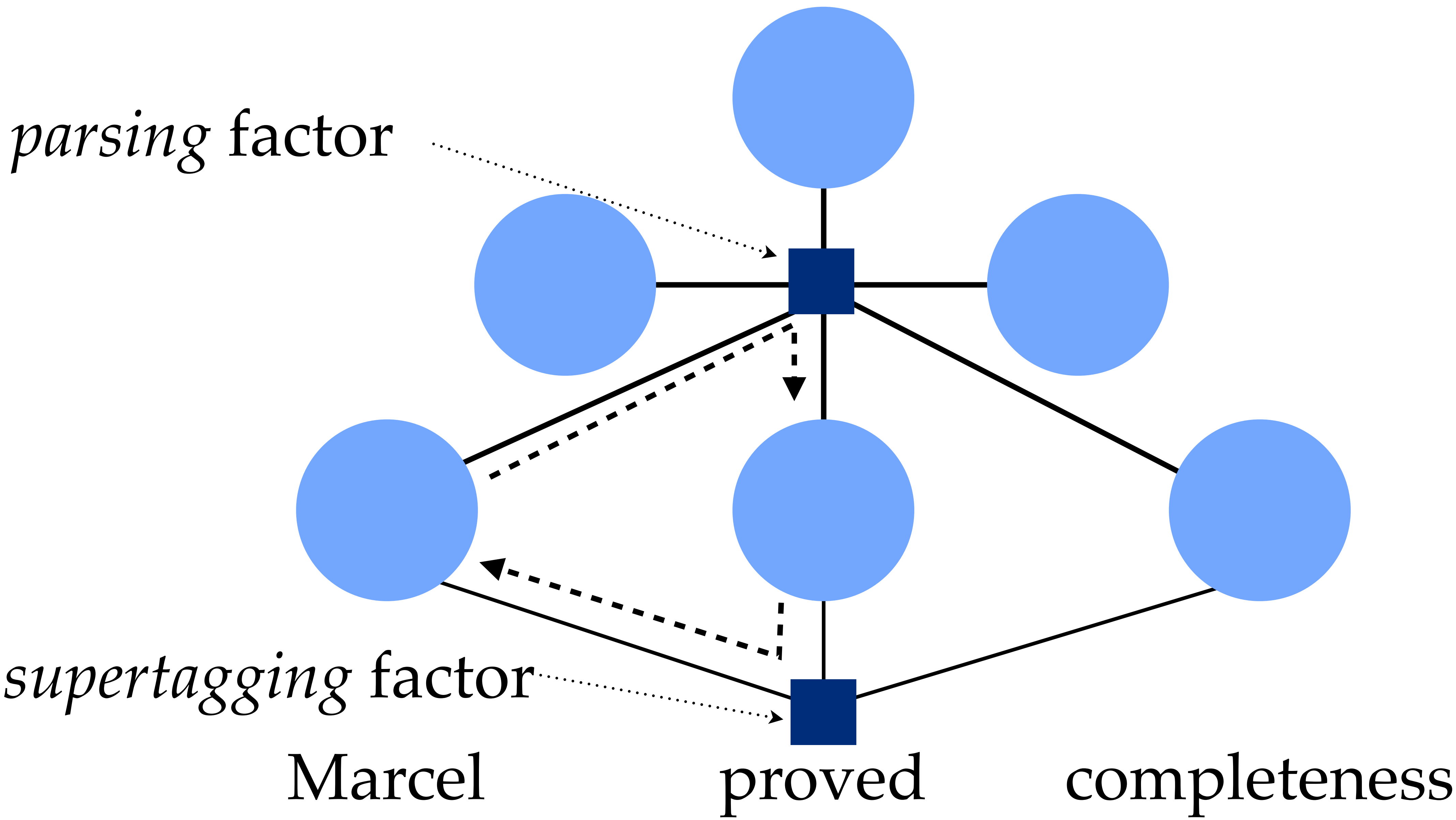
Belief Propagation

Graph is not a tree!



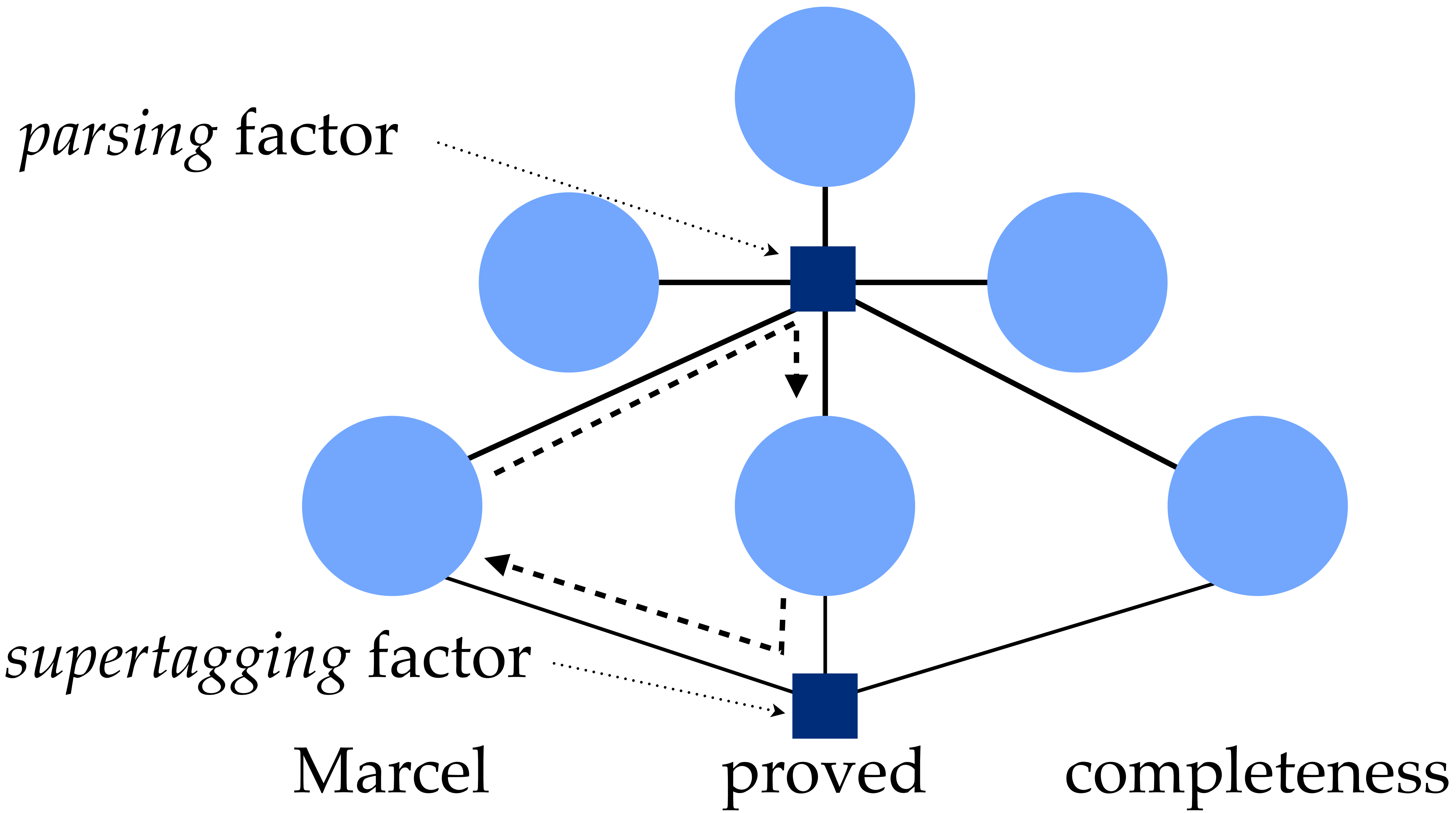
Belief Propagation

Graph is not a tree!



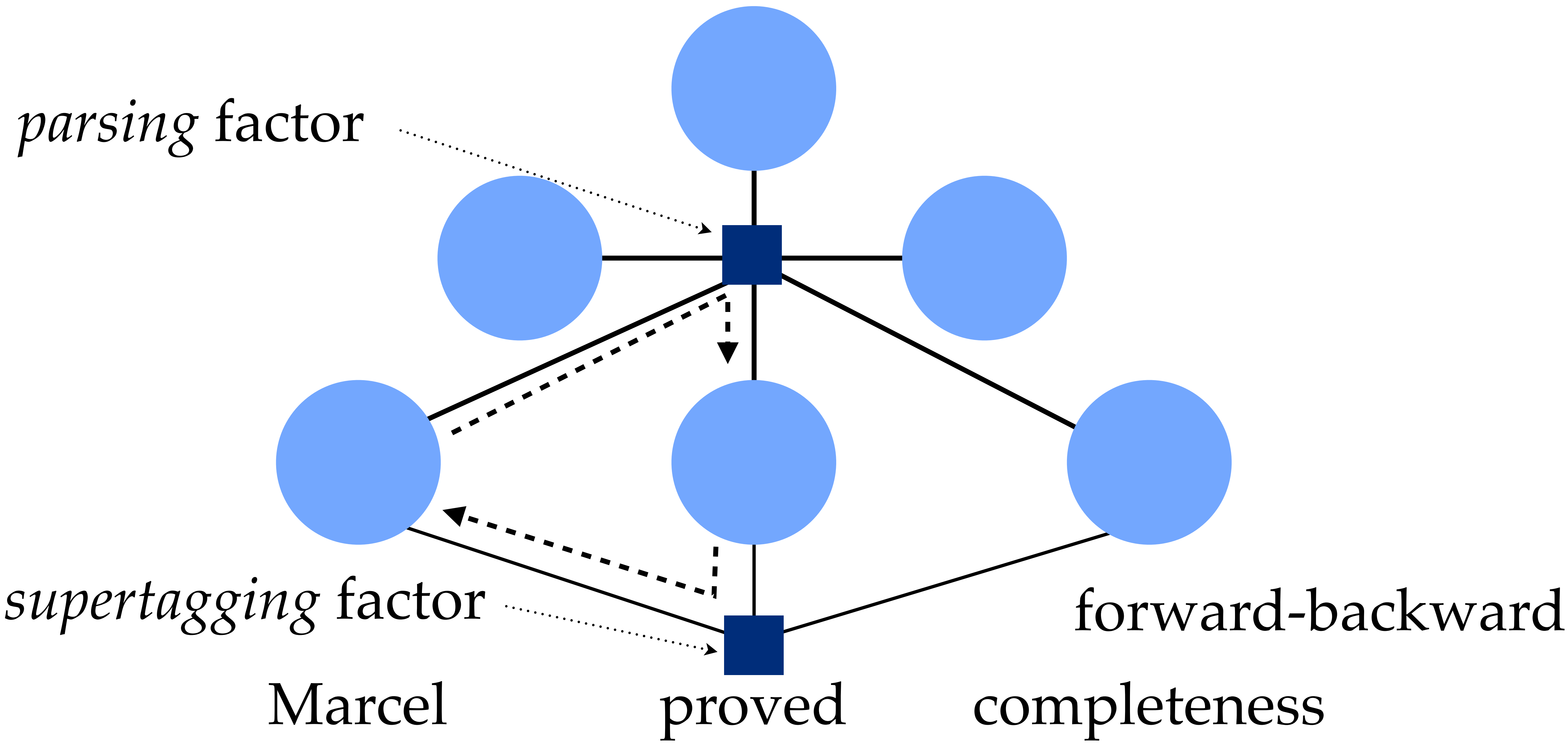
Loopy Belief Propagation

Graph is not a tree!



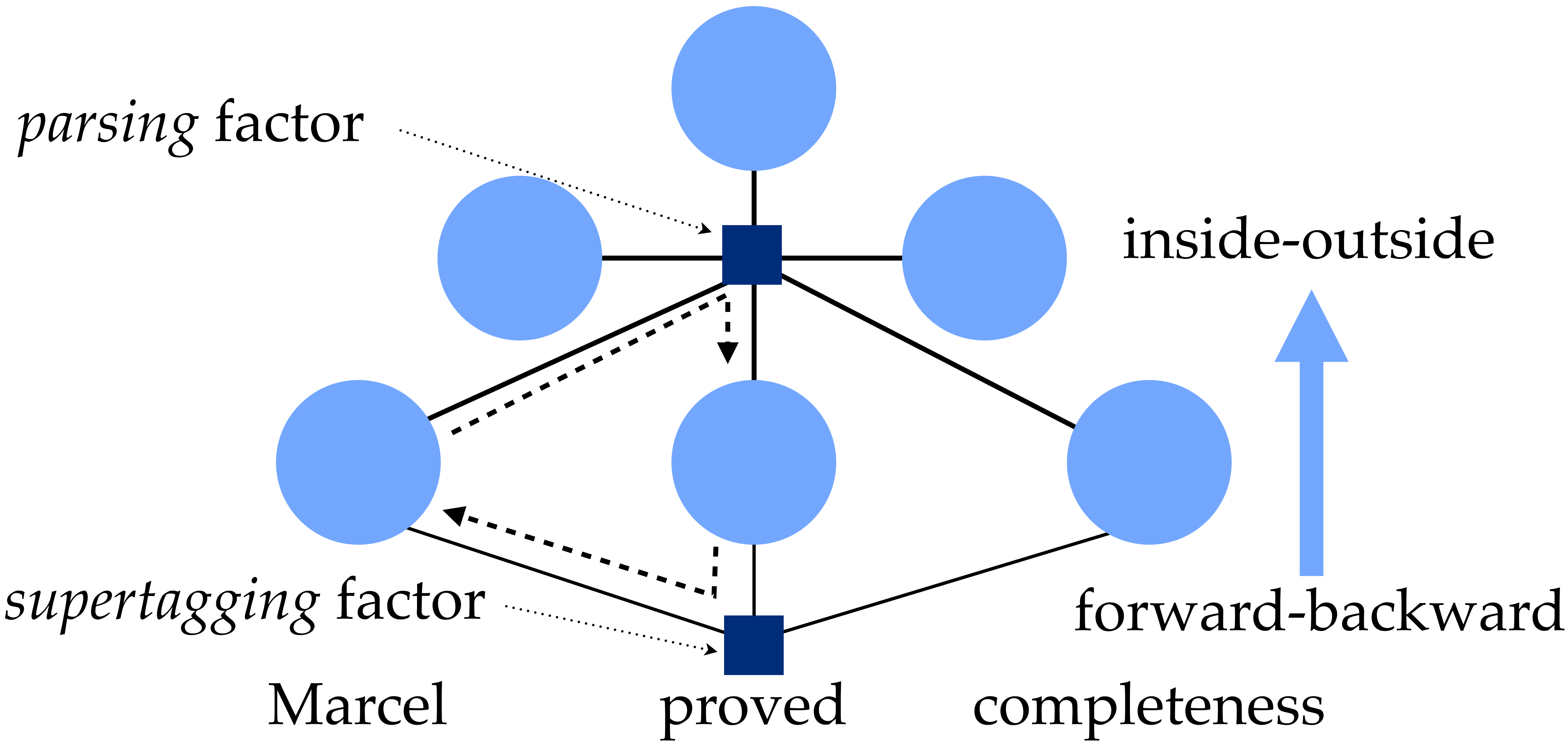
Loopy Belief Propagation

Graph is not a tree!



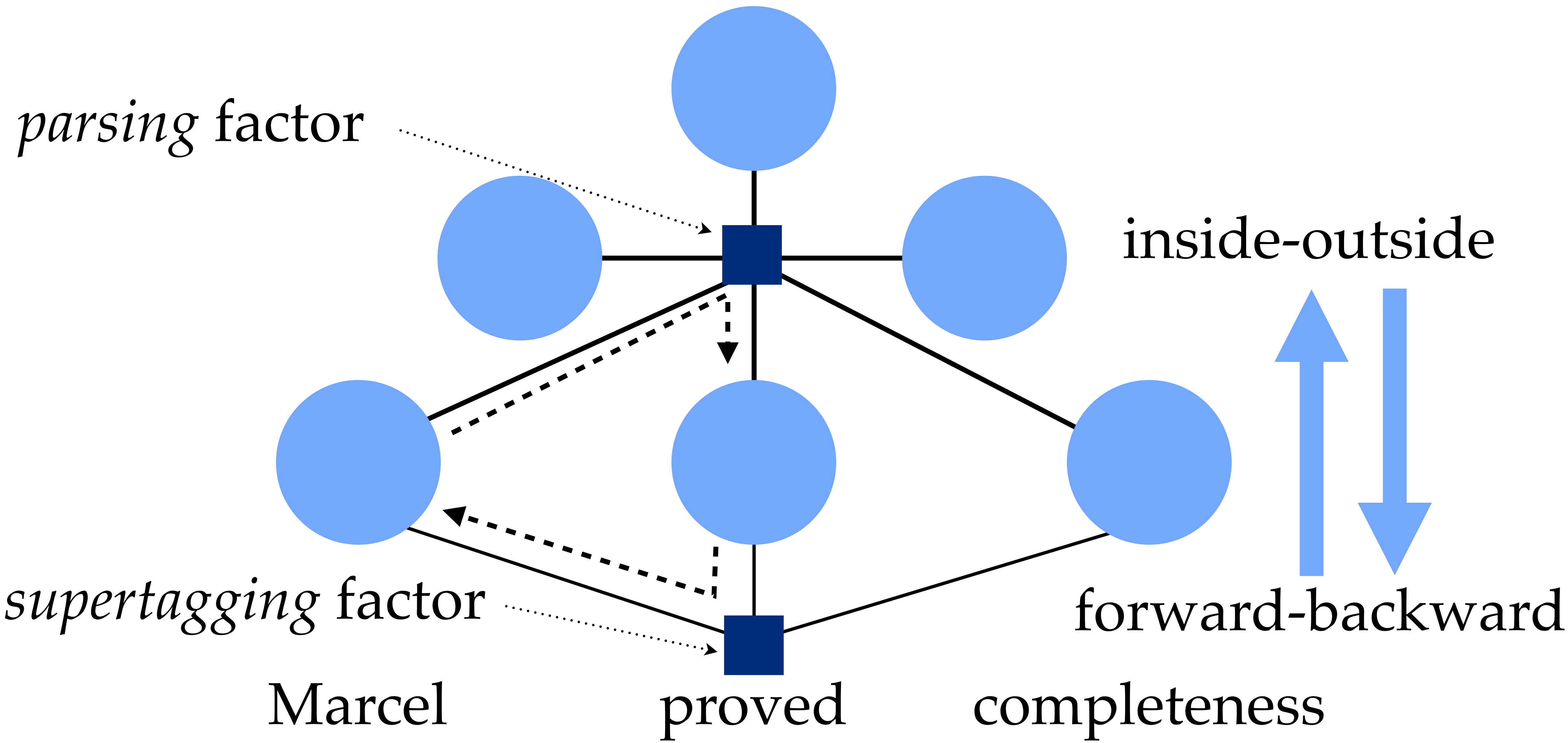
Loopy Belief Propagation

Graph is not a tree!



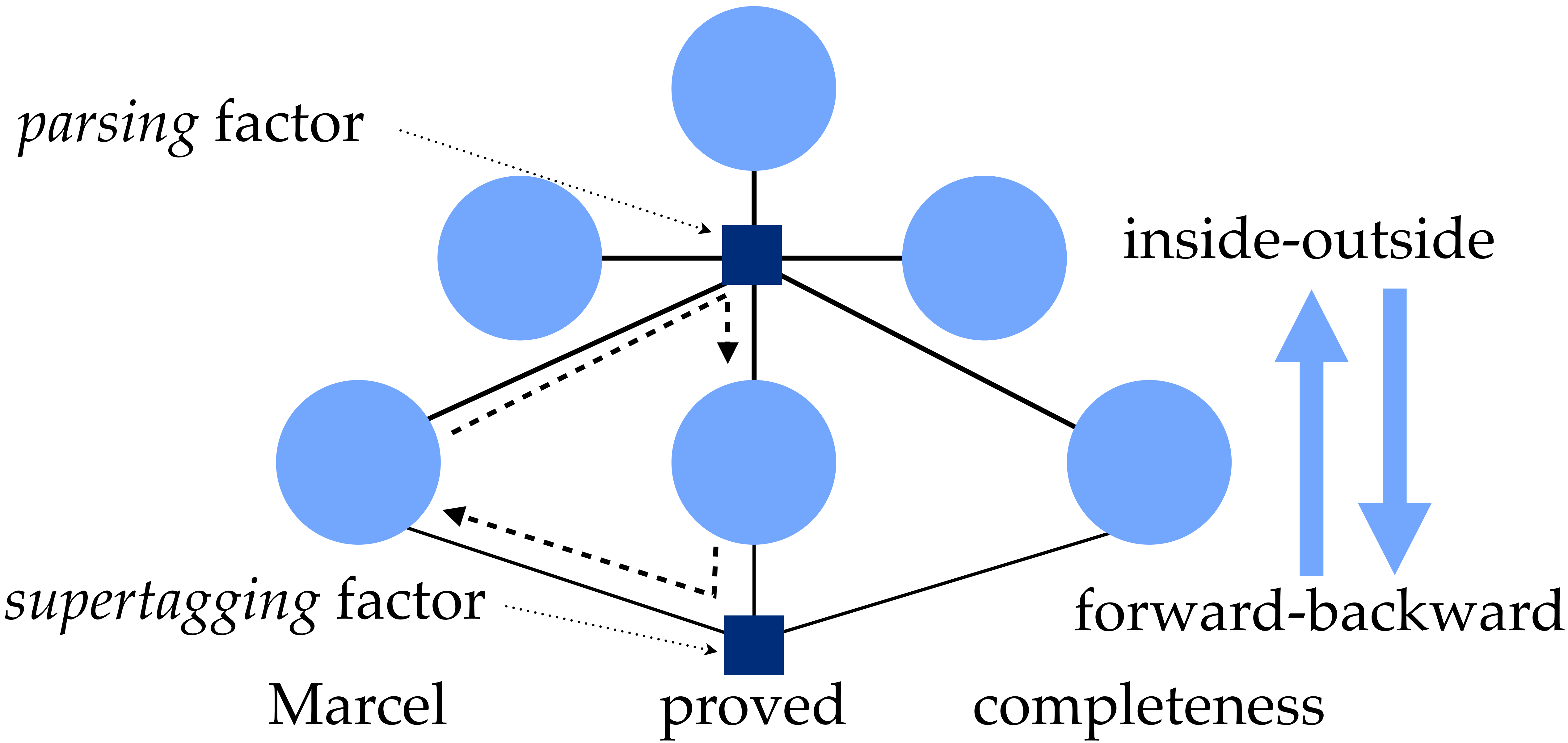
Loopy Belief Propagation

Graph is not a tree!



Loopy Belief Propagation

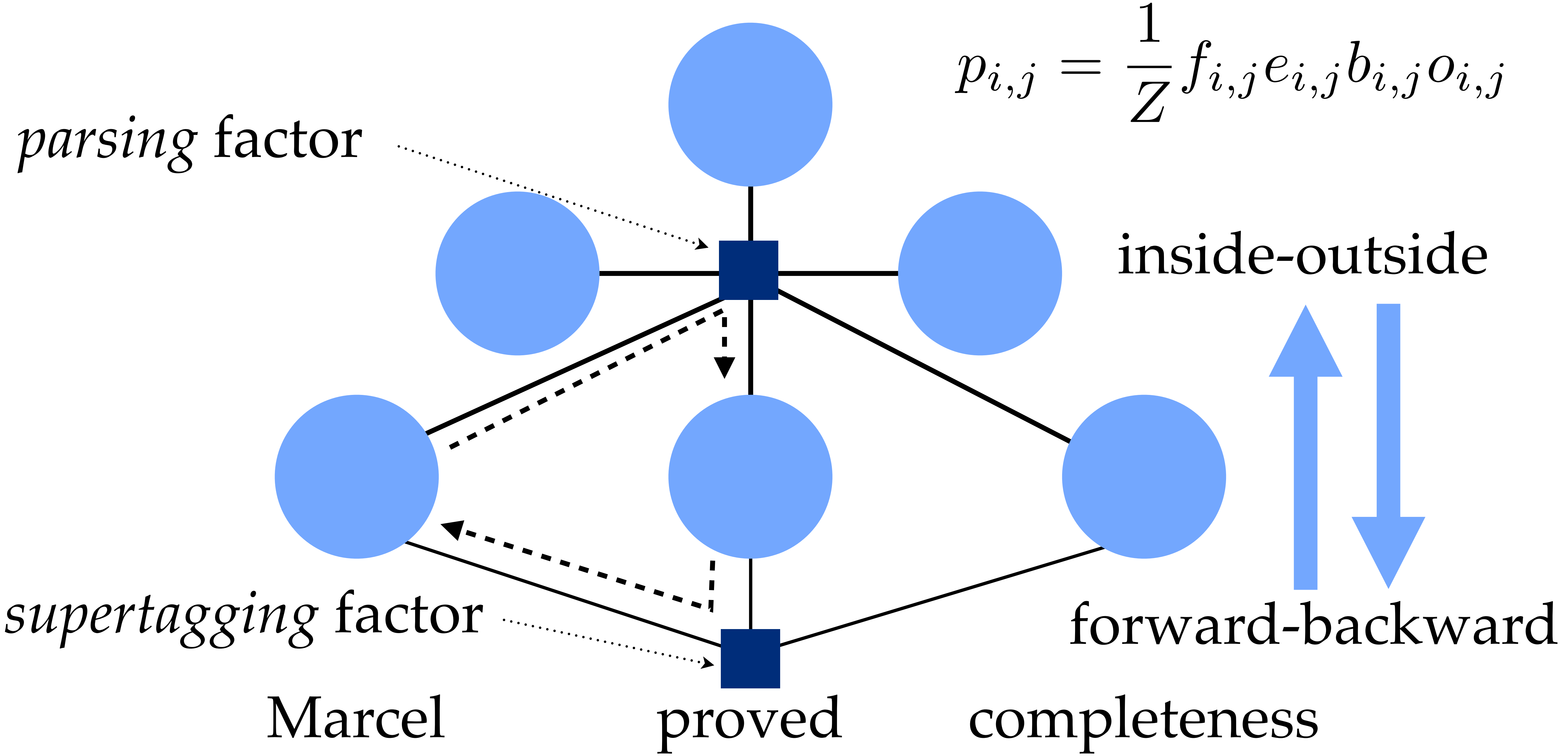
Graph is not a tree!



Converges to bounded approximate marginals (Yedidia et al. 2001)

Loopy Belief Propagation

Graph is not a tree!

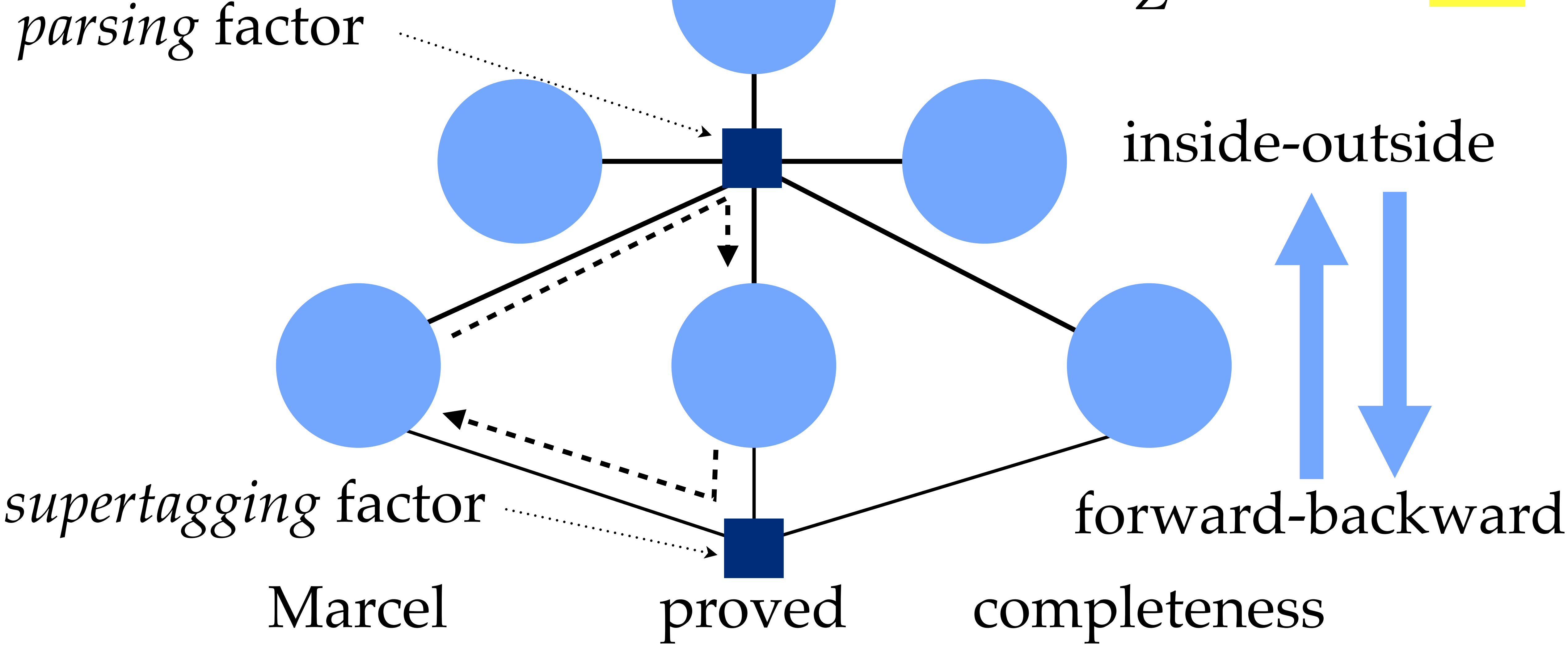


Converges to bounded approximate marginals (Yedidia et al. 2001)

Loopy Belief Propagation

Graph is not a tree!

$$p_{i,j} = \frac{1}{Z} f_{i,j} e_{i,j} b_{i,j} o_{i,j}$$

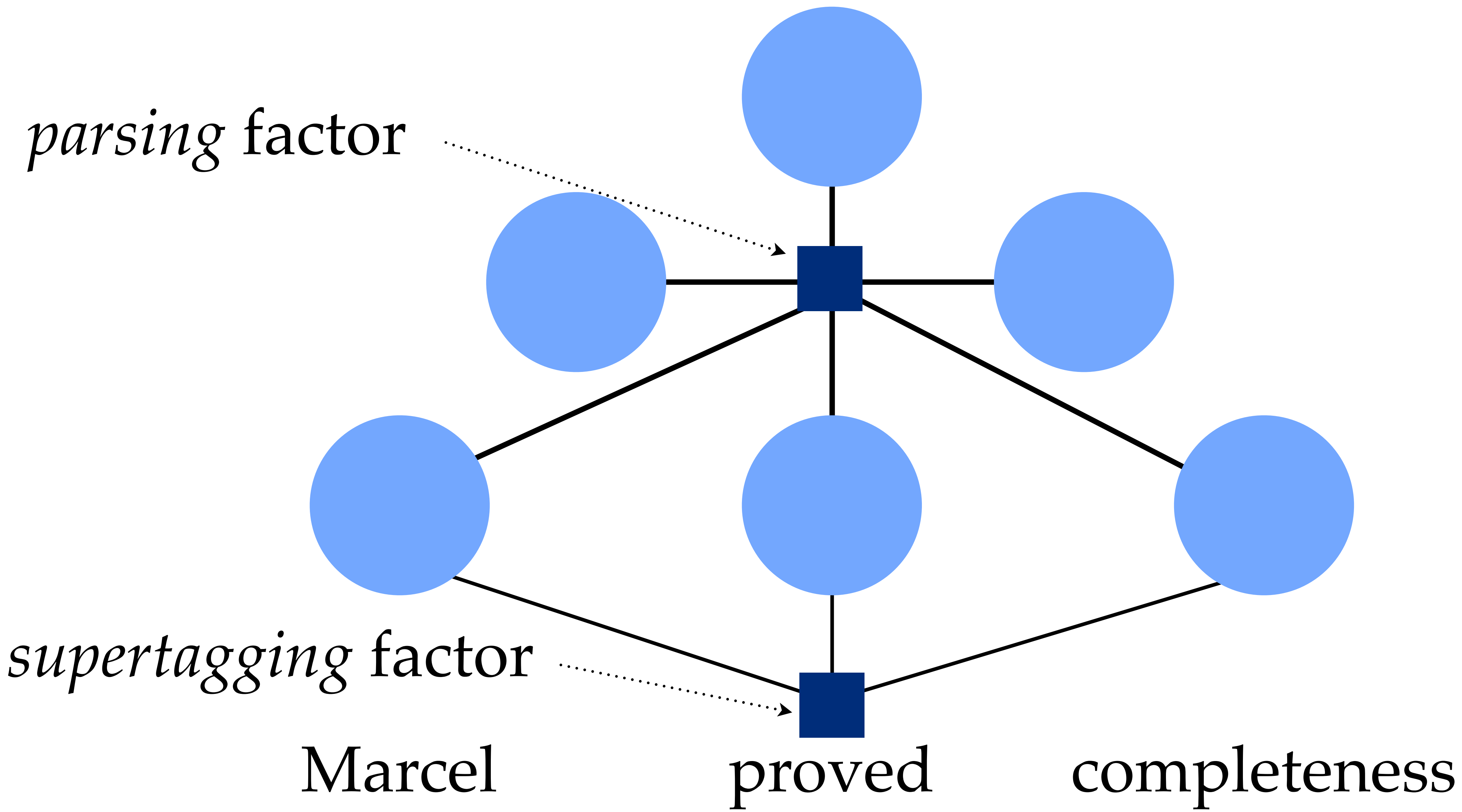


Converges to bounded approximate marginals (Yedidia et al. 2001)

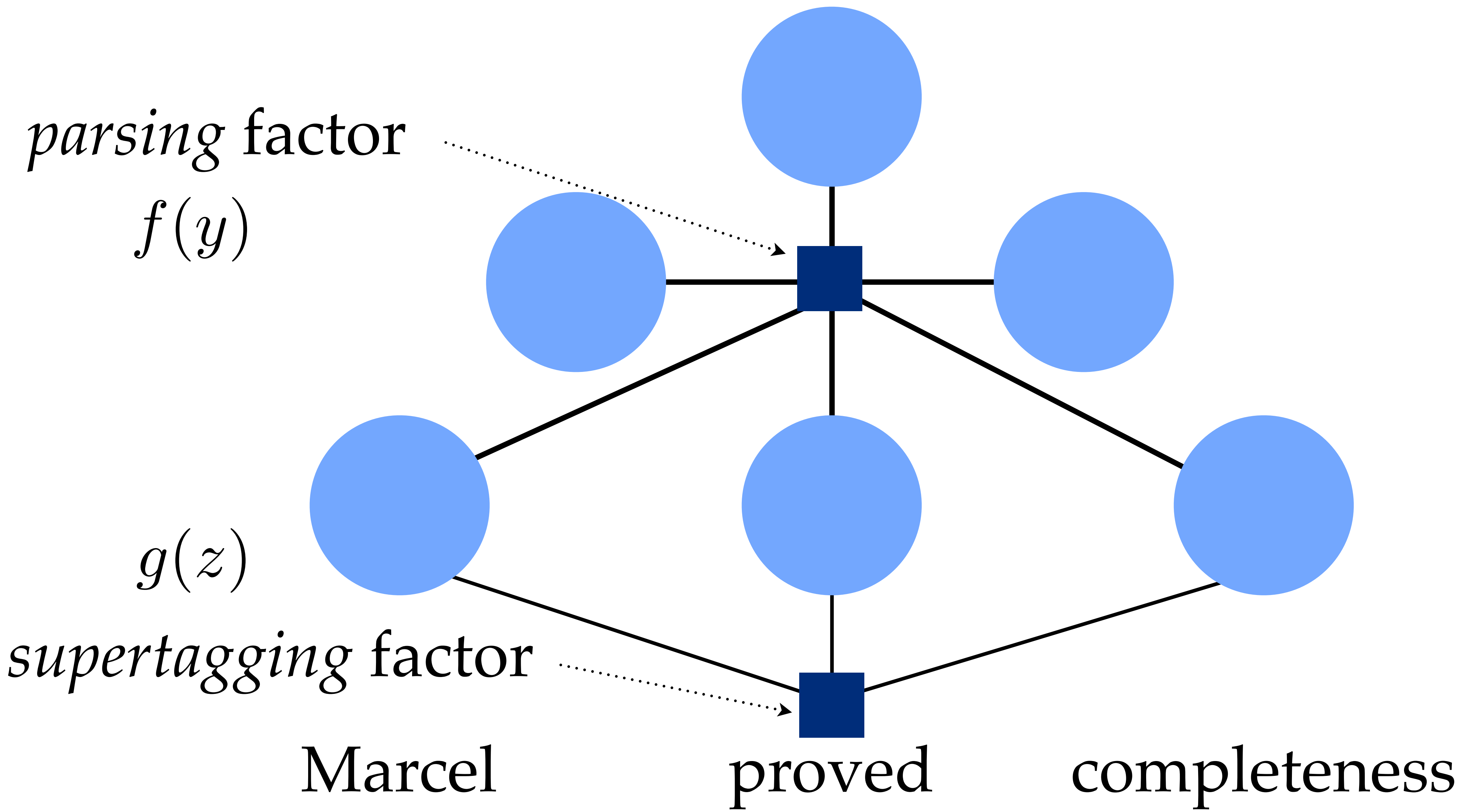
Loopy Belief Propagation

- Computes approximate marginals.
- Complexity is additive: $O(Gn^3 + Gn)$ vs. $O(G^3n^3)$
- In training: use for gradient optimization (e.g. SGD).
- In decoding: compute minimum-risk parse (Goodman 1996).

Dual Decomposition

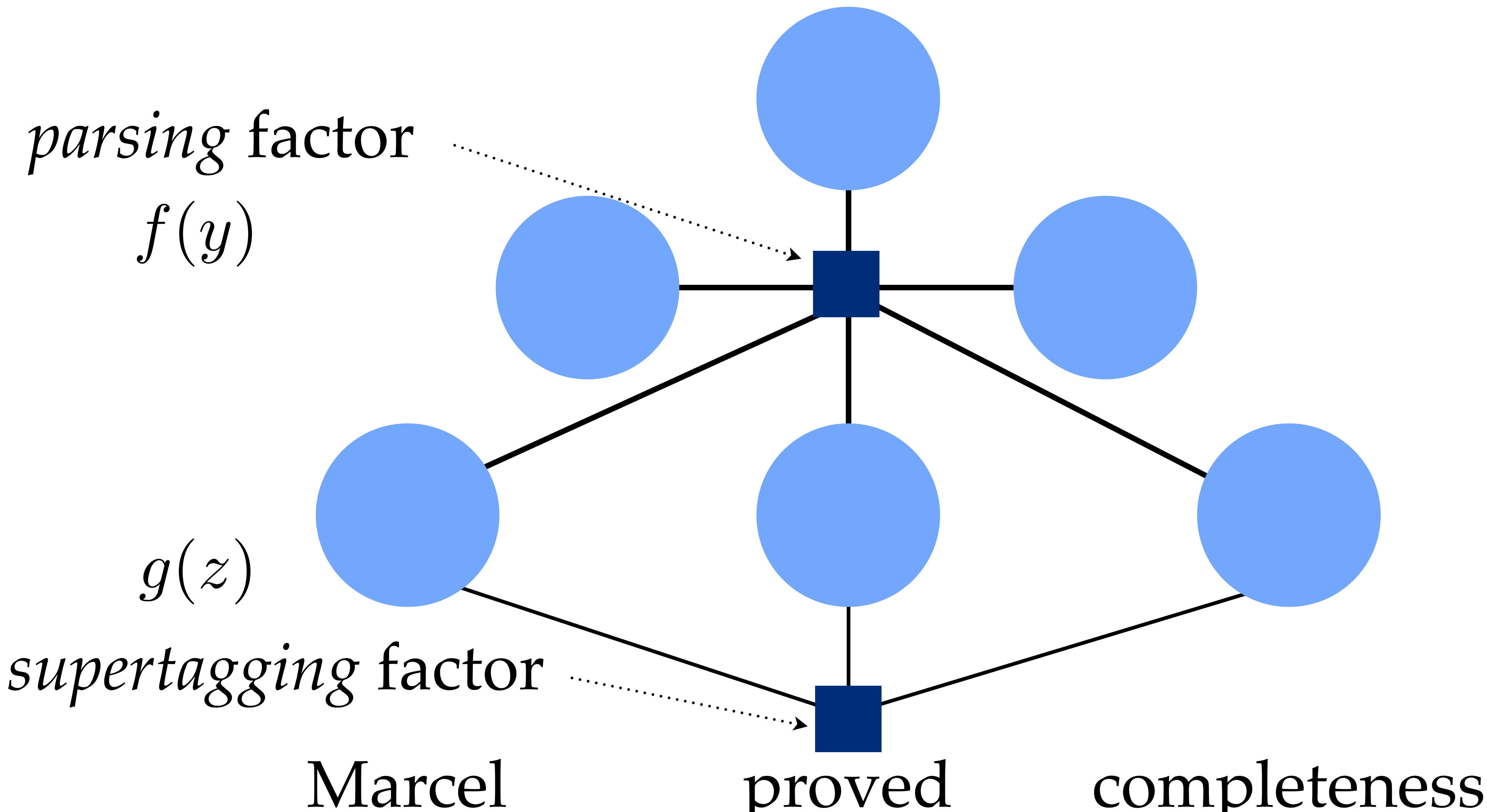


Dual Decomposition



Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \quad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$



Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

$$L(u) = \max_{y} f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$
$$+ \max_{z} g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

$$L(u) = \max_y f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$

relaxed original problem

$$+ \max_z g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

$$L(u) = \max_y f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$

modified subproblem

$$+ \max_z g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

$$L(u) = \max_y f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$
$$+ \max_z g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual objective: find assignment of $u(i,t)$ that minimizes $L(u)$

Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

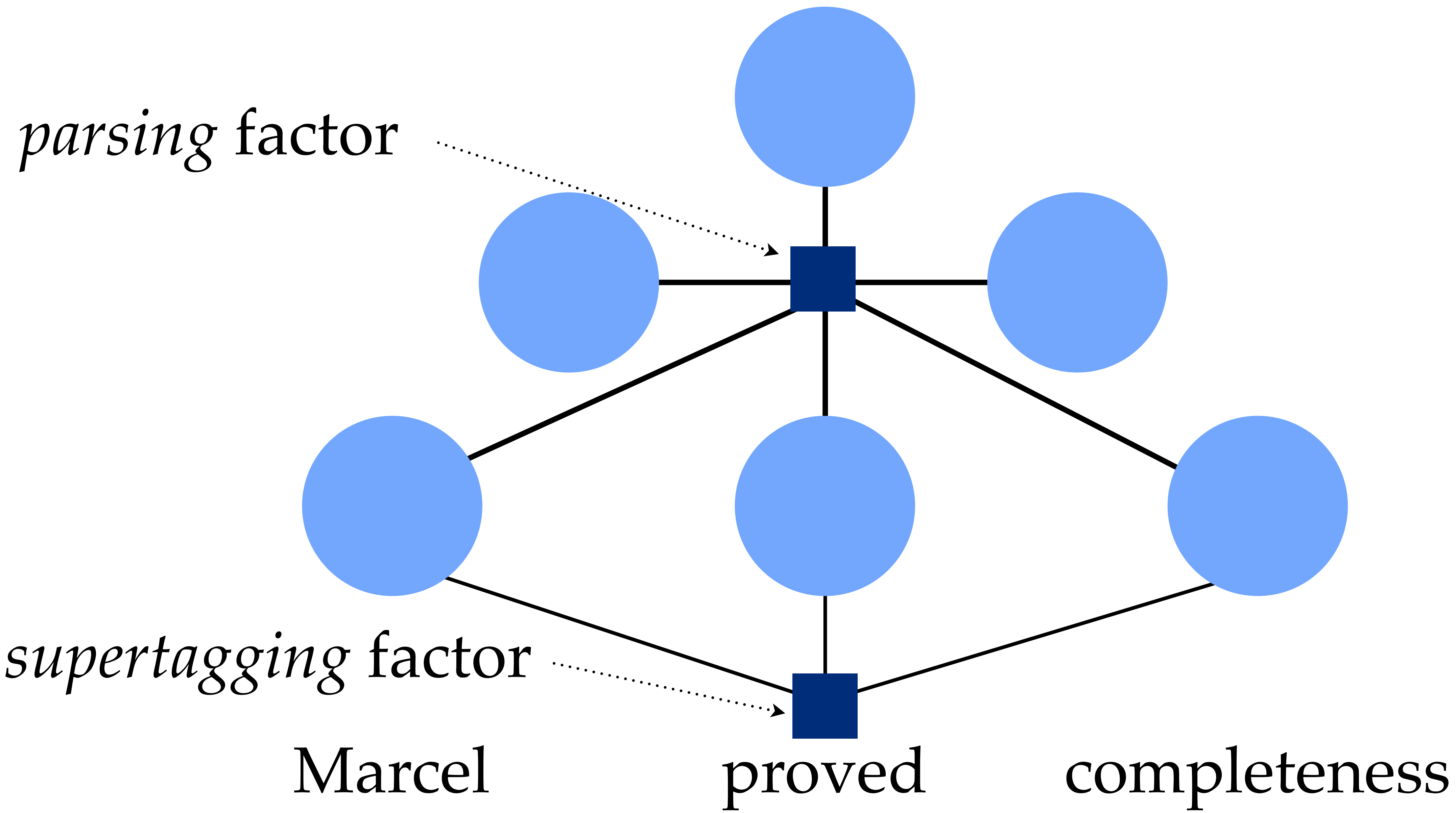
$$L(u) = \max_{y} f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$
$$+ \max_{z} g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual objective: find assignment of $u(i,t)$ that minimizes $L(u)$

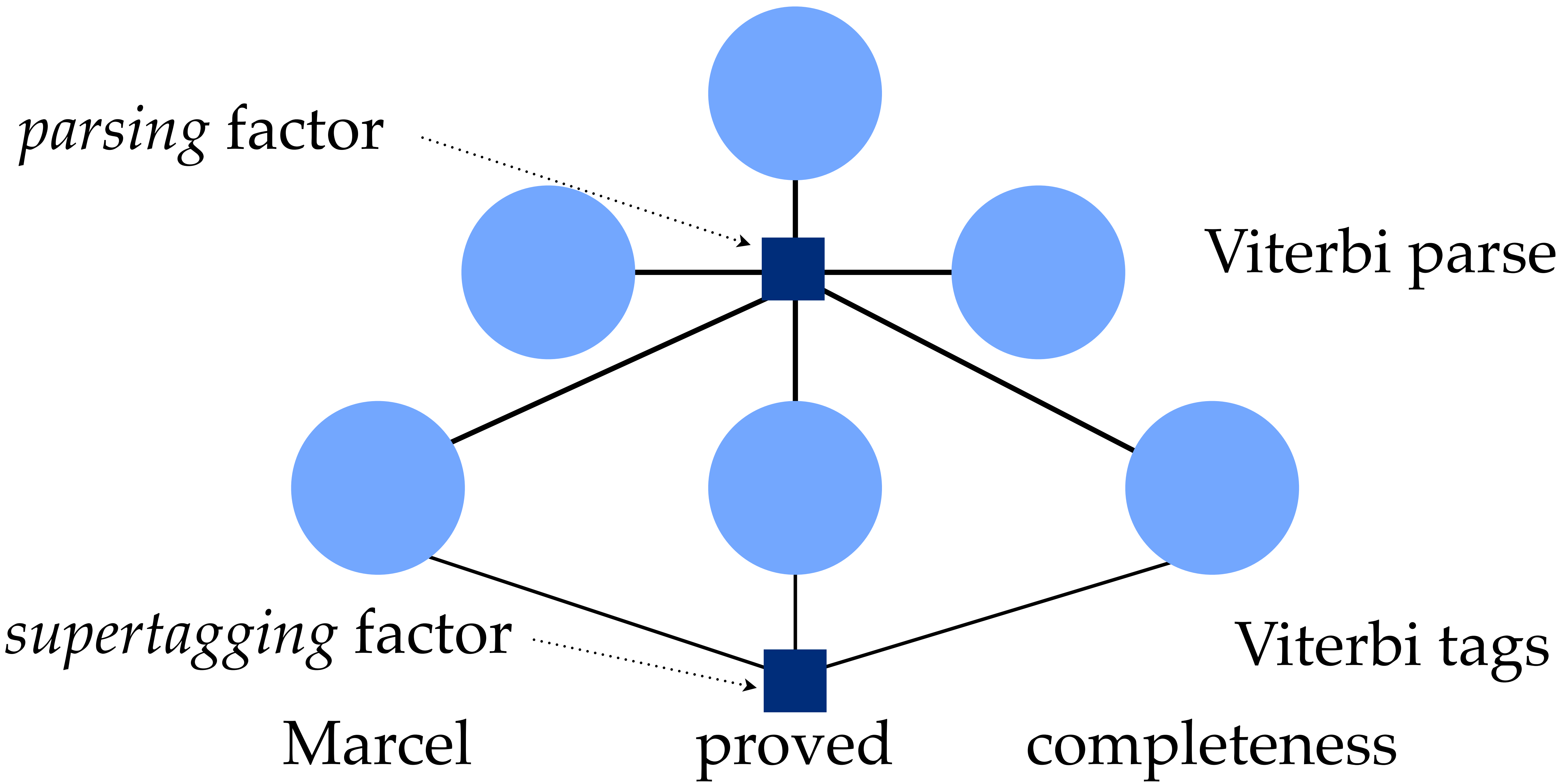
$$u(i,t) = u(i,t) + \alpha \cdot [y(i,t) - z(i,t)]$$ (Rush et al. 2010)

Solution provably solves original problem.

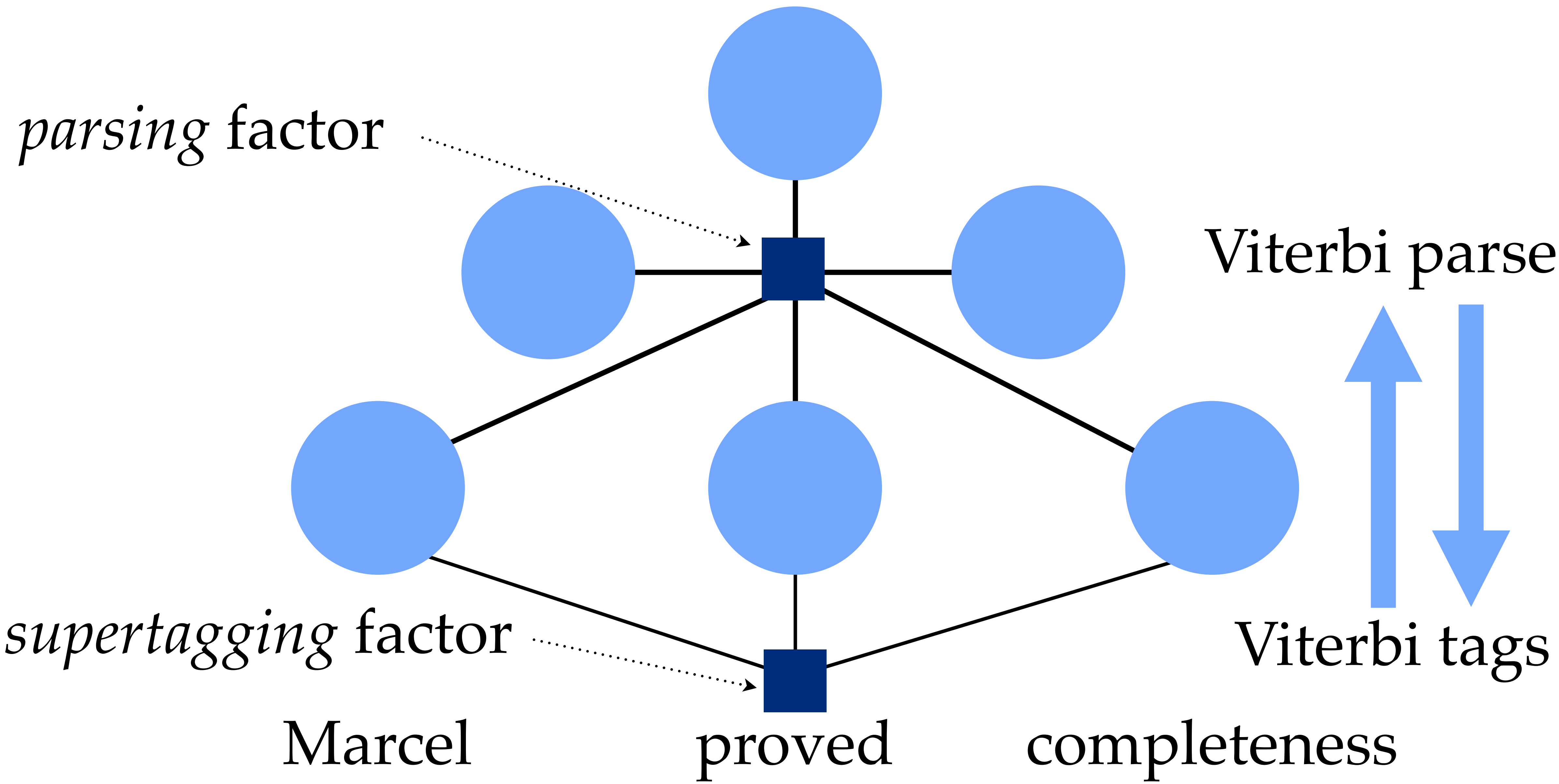
Dual Decomposition



Dual Decomposition



Dual Decomposition



Dual Decomposition

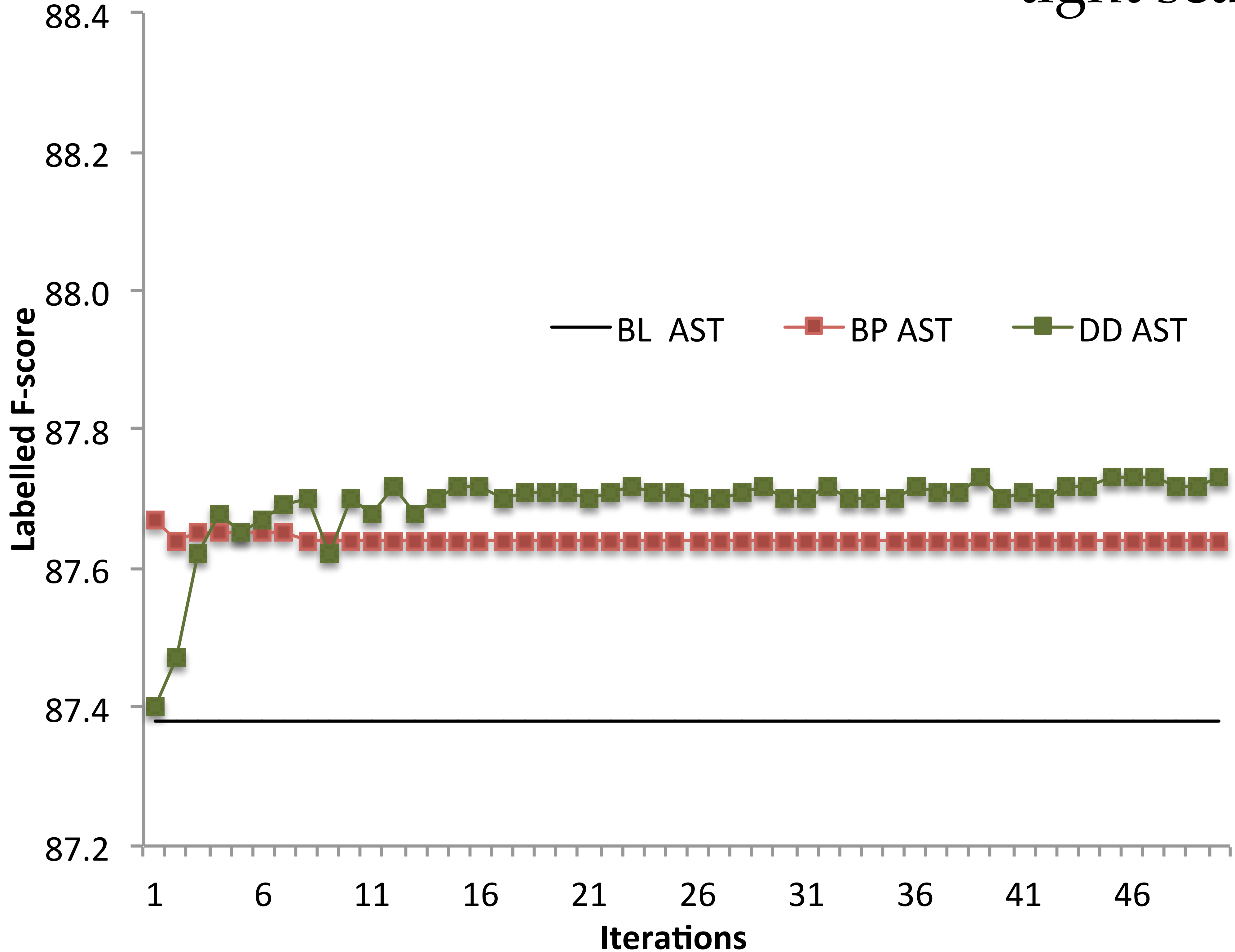
- Computes *exact* maximum, *if* it converges.
  - Otherwise: return best parse seen (approximation).
- Complexity is additive: $O(Gn^3 + Gn)$ vs. $O(G^3n^3)$
- In training: use with margin-based optimizers.
- In decoding: compute Viterbi parse.

Experiments

- Standard parsing task:
  - C&C Parser and supertagger (Clark & Curran 2007).
  - CCGBank standard train/dev/test splits.
  - Separate L-BFGS optimization for each submodel (pseudolikelihood: Besag 1975).
  - Approximate algorithms used to decode test set *only*.

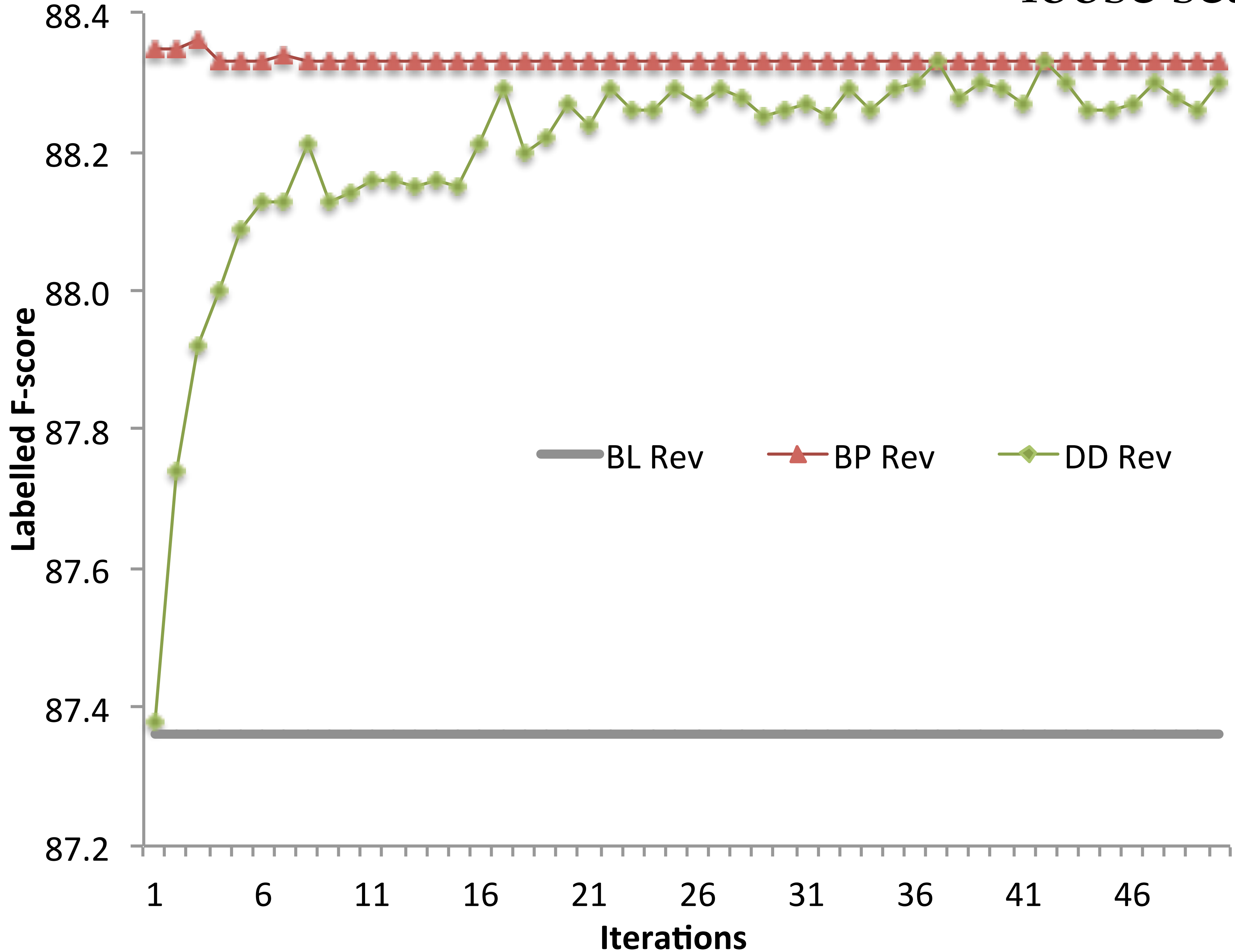
Experiments

tight search (AST)

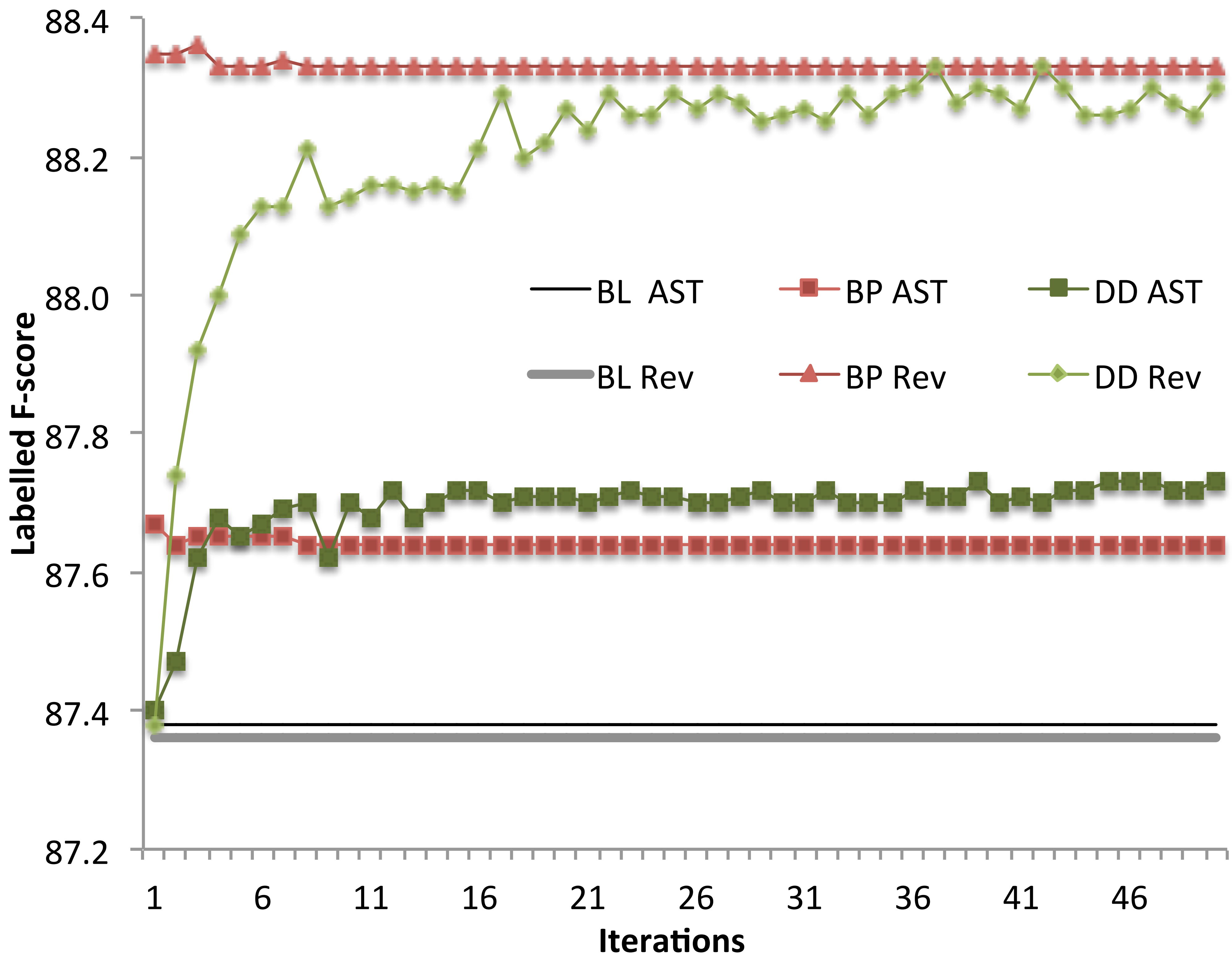


Experiments

loose search (Rev)



Experiments



Experiments

Experiments

- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65

Experiments

- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65
- Belief propagation (1 iter):
  - tight beam: 88.25
  - loose beam: **88.86**

Experiments

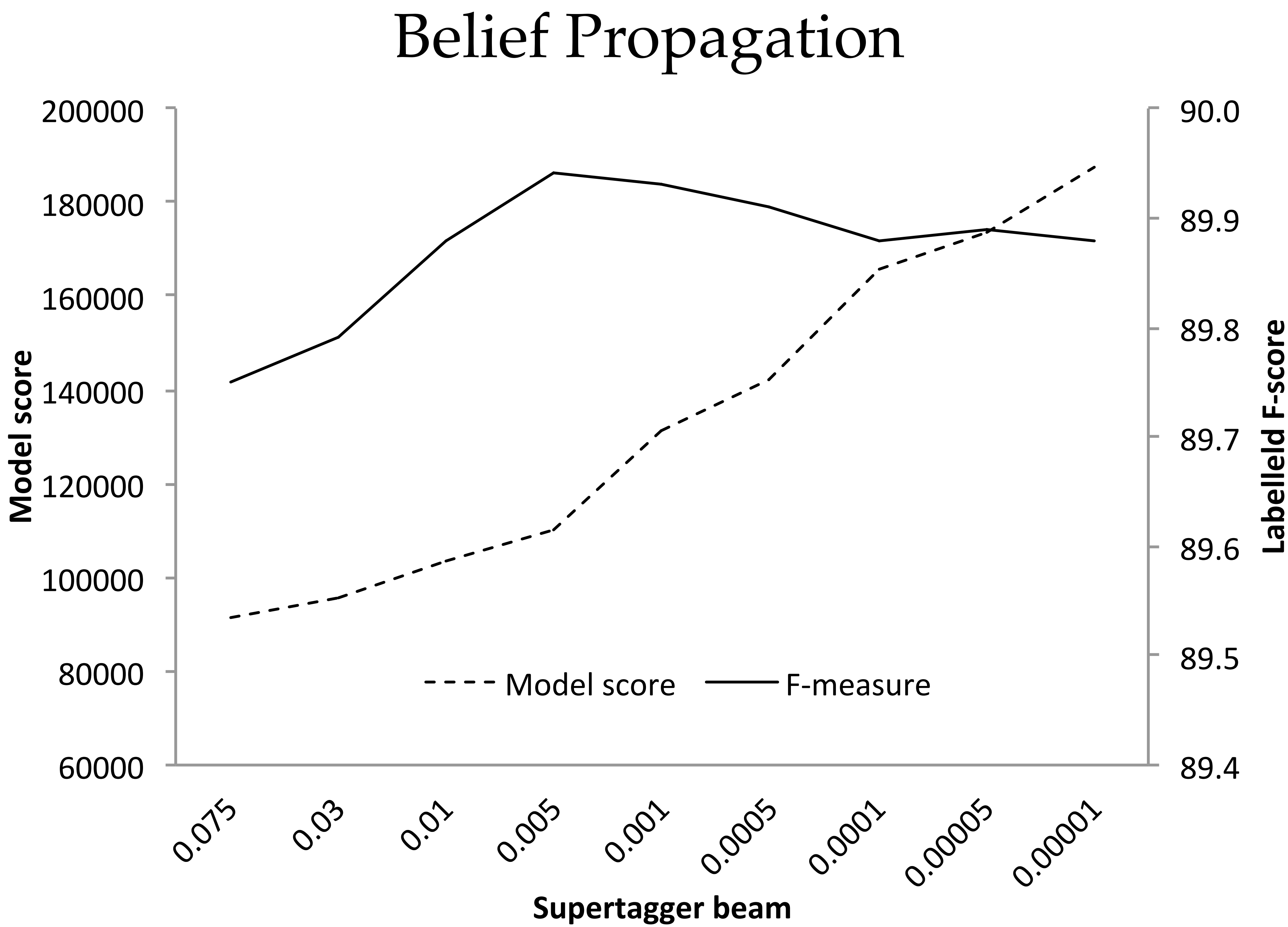
- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65
- Belief propagation (1 iter):
  - tight beam: 88.25
  - loose beam: **88.86**
- Dual decomposition (25 iter):
  - tight beam: 88.14
  - loose beam: **88.80**

Experiments

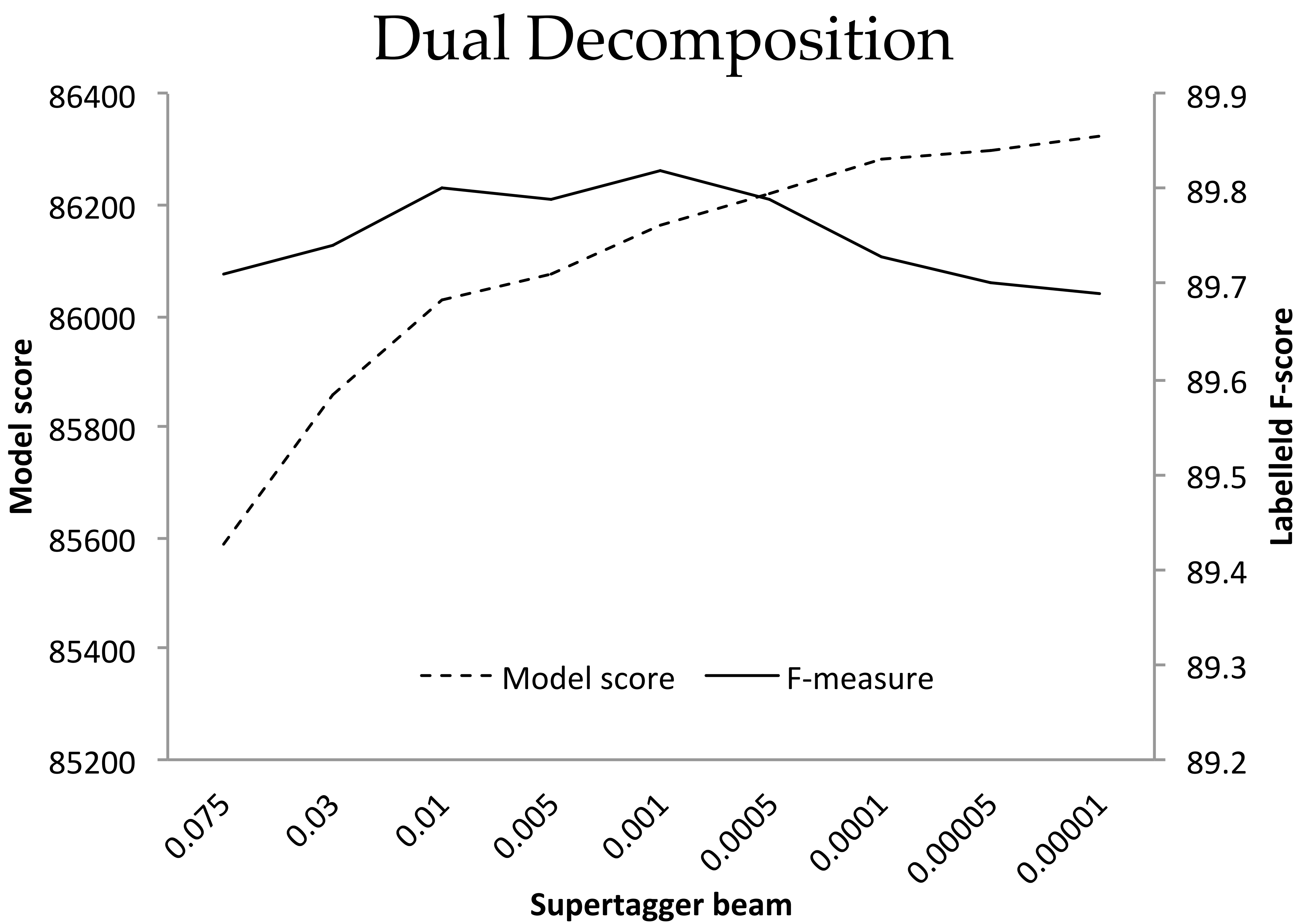
- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65
- Belief propagation (1 iter):
  - tight beam: 88.25
  - loose beam: **88.86**
- Dual decomposition (25 iter):
  - tight beam: 88.14
  - loose beam: **88.80**

Best performance
in larger search space

Oracle Results Again



Oracle Results Again



Summary

- Parser and supertagger interaction exploited in combined model.

Summary

- Parser and supertagger interaction exploited in combined model.
- By far best performance with loose supertagger beam.
  Better models can exploit larger search spaces.

Summary

- Parser and supertagger interaction exploited in combined model.
- By far best performance with loose supertagger beam.
  Better models can exploit larger search spaces.
- Accurate parsing possible in a combined model.

Overview

- The Problem with Pipeline Models
- Decoding Integrated Models
  with Loopy Belief Propagation and Dual Decomposition (ACL 2011)
- Training Integrated Models
  with Softmax-Margin using Exact and Approximate Loss Functions (EMNLP 2011)

Softmax-Margin

- Minimizes bound on expected risk (Gimpel & Smith, 2010).
- Allows optimization towards loss function.
- Retains probabilistic interpretation.
- Convex.
- Requires little change to existing Conditional Log-Likelihood (CLL) implementation.

Training Objectives

CLL: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

Training Objectives

weights

CLL: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

Training Objectives

weights features

CLL:
$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

Training Objectives

CLL:

$$\min_{\theta} \sum_{i=1}^{m} \left[-\underbrace{\theta}_{\text{weights}}^{\mathsf{T}} \underbrace{f}_{\text{features}}(\underbrace{x^{(i)}}_{\text{input}}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

Training Objectives

weights features input true output

CLL: $$\min_\theta \sum_{i=1}^{m} \left[-\theta^\mathsf{T} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^\mathsf{T} f(x^{(i)}, y)\} \right]$$

Training Objectives

weights features input true output proposed output

CLL:

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

Training Objectives

weights features input true output proposed output

CLL: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

SMM: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right]$$

Training Objectives

weights features input true output proposed output

CLL: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right]$$

SMM: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right]$$

Training Objectives

CLL:

$$\min_{\theta} \sum_{i=1}^{m} \left[-\overset{\text{weights}}{\theta}^{\top} \overset{\text{features}}{f}(\overset{\text{input}}{x^{(i)}}, \overset{\text{true output}}{y^{(i)}}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\top} f(x^{(i)}, \overset{\text{proposed output}}{y})\} \right]$$

SMM:

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\top} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\top} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right]$$

- Penalise high-loss outputs.
- Re-weight outcomes by *loss*.
- Loss function an *unweighted feature* -- if **decomposable.**

Parsing Metrics

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

$$
\begin{array}{ccc}
\text{Marcel} & \text{proved} & \text{completeness} \\
\hline
NP & (S\backslash NP)/NP & NP \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP \quad >} \\
\hline
\multicolumn{3}{c}{S \quad <}
\end{array}
$$

Parsing Metrics

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Precision $$P(y, y') = \frac{|y \cap y'|}{|y'|} = \frac{n}{d}$$

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Precision $$P(y, y') = \frac{|y \cap y'|}{|y'|} = \frac{n}{d}$$

Recall $$R(y, y') = \frac{|y \cap y'|}{|y|} = \frac{n}{|y|}$$

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Precision $$P(y, y') = \frac{|y \cap y'|}{|y'|} = \frac{n}{d}$$

Recall $$R(y, y') = \frac{|y \cap y'|}{|y|} = \frac{n}{|y|}$$

F-measure $$F_1(y, y') = \frac{2PR}{P + R} = \frac{2|y \cap y'|}{|y| + |y'|} = \frac{2n}{d + |y|}$$

Decomposability

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Decomposability

$$|y \cap y'| = n \qquad \text{correct dependencies returned}$$
$$|y'| = d \qquad \text{all dependencies returned}$$

- e.g. n is decomposable $\qquad n_1 + n_2 = n$

Decomposability

$$|y \cap y'| = n \qquad \text{correct dependencies returned}$$
$$|y'| = d \qquad \text{all dependencies returned}$$

- e.g. n is decomposable $n_1 + n_2 = n$
- But F-measure is *not* $f_1 + f_2 \neq f$

Decomposability

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

- e.g. n is decomposable $n_1 + n_2 = n$
- But F-measure is *not* $f_1 + f_2 \neq f$

... but statistics are decomposable! $F_1(y, y') = \frac{2n}{d + |y|}$

Decomposability

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

- e.g. n is decomposable $n_1 + n_2 = n$
- But F-measure is *not* $f_1 + f_2 \neq f$

... but statistics are decomposable! $F_1(y, y') = \frac{2n}{d + |y|}$

$$f_1 + f_2 = \frac{2n_1}{d_1 + |y_1|} \oplus \frac{2n_2}{d_2 + |y_2|} = \frac{2(n_1 + n_2)}{d_1 + d_2 + |y_1| + |y_2|}$$

Decomposable Losses for Parsing

Decomposable Losses for Parsing

$T(y)$ = set of actions to build parse y

$$\frac{\mathrm{NP} \quad \mathrm{S\backslash NP}}{\mathrm{S}} <$$

Decomposable Losses for Parsing

$T(y)$ = set of actions to build parse y $\qquad \dfrac{\text{NP} \quad \text{S}\backslash\text{NP}}{\text{S}} <$

$d_+(t)$ = number of dependencies introduced by $t \in T(y)$

$n_+(t)$ = number of correct dependencies introduced by $t \in T(y)$

Decomposable Losses for Parsing

$T(y)$ = set of actions to build parse y

$$\frac{\text{NP} \quad \text{S}\backslash\text{NP}}{\text{S}} <$$

$d_+(t)$ = number of dependencies introduced by $t \in T(y)$

$n_+(t)$ = number of correct dependencies introduced by $t \in T(y)$

Precision $$DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t)$$

Decomposable Losses for Parsing

$T(y)$ = set of actions to build parse y $\qquad \dfrac{\text{NP} \quad \text{S}\backslash\text{NP}}{\text{S}} <$

$d_+(t)$ = number of dependencies introduced by $t \in T(y)$

$n_+(t)$ = number of correct dependencies introduced by $t \in T(y)$

Precision $\quad DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t)$

Recall $\quad DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t)$

Decomposable Losses for Parsing

$T(y)$ = set of actions to build parse y $\qquad \dfrac{\text{NP} \quad \text{S}\backslash\text{NP}}{\text{S}} <$

$d_+(t)$ = number of dependencies introduced by $t \in T(y)$

$n_+(t)$ = number of correct dependencies introduced by $t \in T(y)$

Precision $\qquad DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t)$

Recall $\qquad DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t)$

F_1 $\qquad DecF1(y) = DecP(y) + DecR(y)$

(Taskar et al., 2004)

Decomposable Losses for Parsing

$$\frac{NP \quad S\backslash NP}{S} <$$

$T(y)$ = set of actions to build parse y

$d_+(t)$ = number of dependencies introduced by $t \in T(y)$

$n_+(t)$ = number of correct dependencies introduced by $t \in T(y)$

Precision $$DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t)$$

Recall $$DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t)$$

efficient but *approximate!*

F_1 $$DecF1(y) = DecP(y) + DecR(y)$$

(Taskar et al., 2004)

Approximate Losses with CKY

items $A_{i,j}$

Approximate Losses with CKY

items $A_{i,j}$

time_1 flies_2 like_3 an_4 arrow_5

Approximate Losses with CKY

items $A_{i,j}$

target analysis

time_1 flies_2 like_3 an_4 arrow_5

Approximate Losses with CKY

items $A_{i,j}$

target analysis
correct dependencies
all dependencies

time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Approximate Losses with CKY

items $A_{i,j}$

target analysis
correct dependencies
all dependencies

$((S\backslash NP)\backslash(S\backslash NP))/NP_{2,3}$

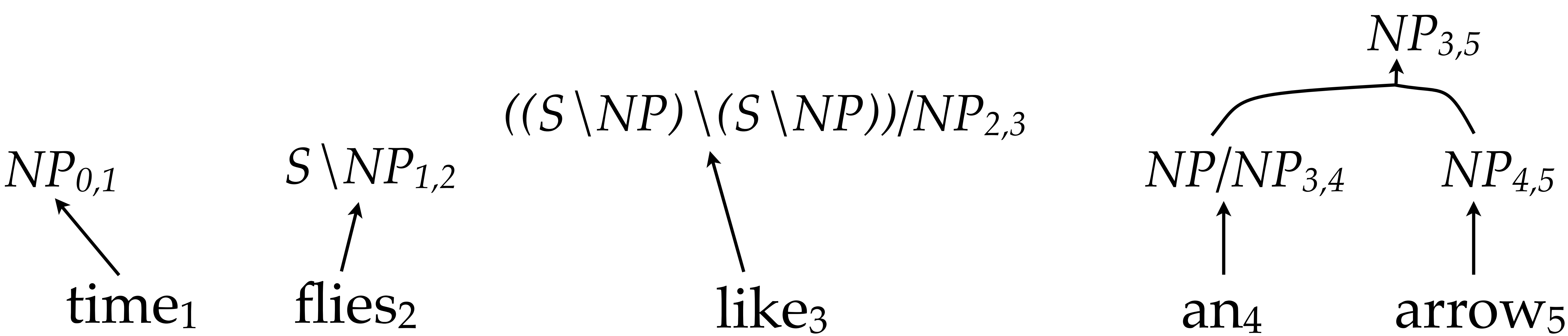
$NP_{0,1}$ $S\backslash NP_{1,2}$ $NP/NP_{3,4}$ $NP_{4,5}$

time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Approximate Losses with CKY

items $A_{i,j}$

target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

target analysis
correct dependencies
all dependencies

$$NP_{3,5}$$

$$\mathrm{DecF_1}(1,1)$$

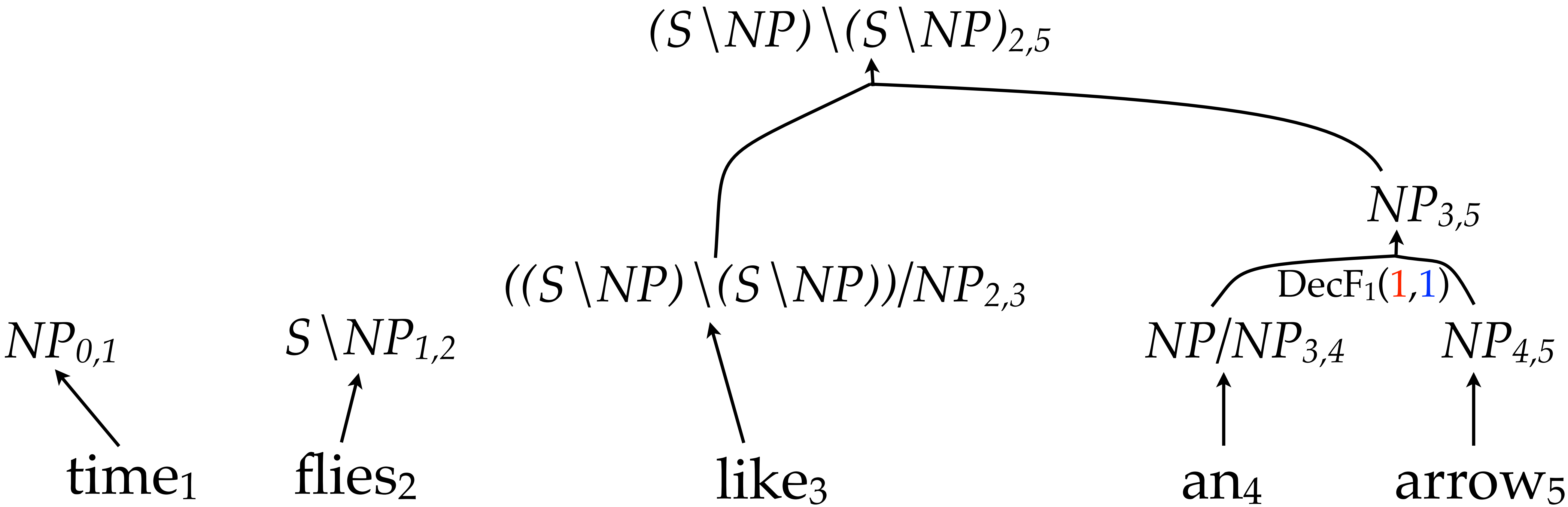
$NP_{0,1}$ $S\backslash NP_{1,2}$ $((S\backslash NP)\backslash(S\backslash NP))/NP_{2,3}$ $NP/NP_{3,4}$ $NP_{4,5}$

time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Approximate Losses with CKY

items $A_{i,j}$

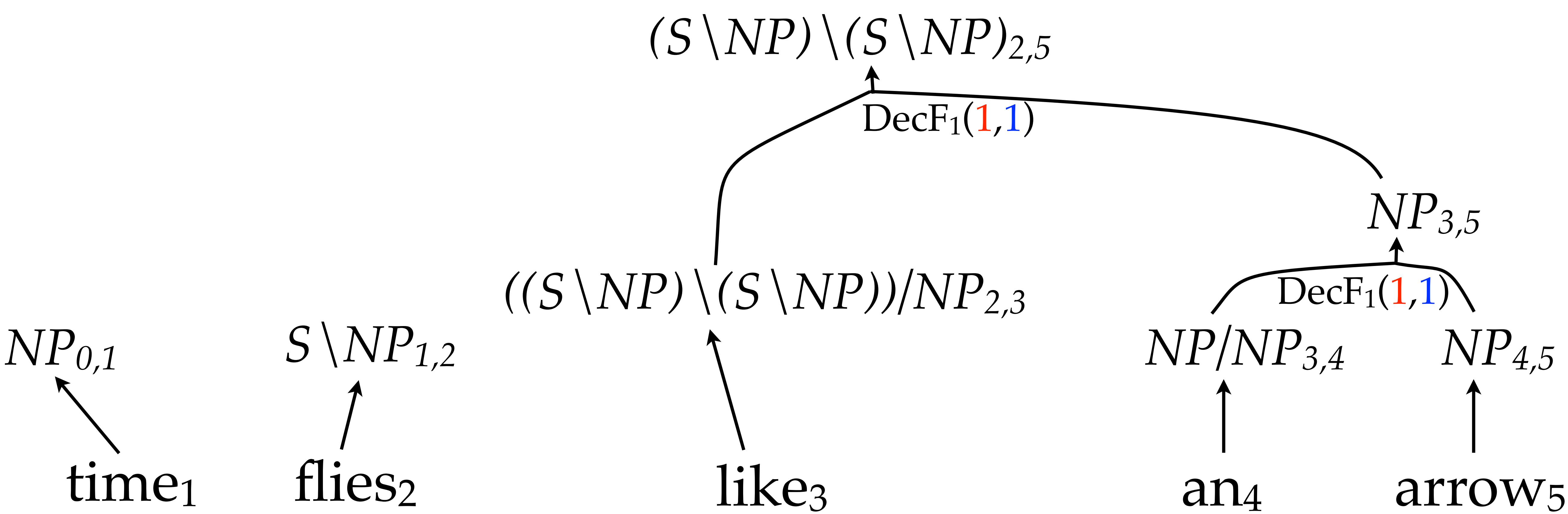
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

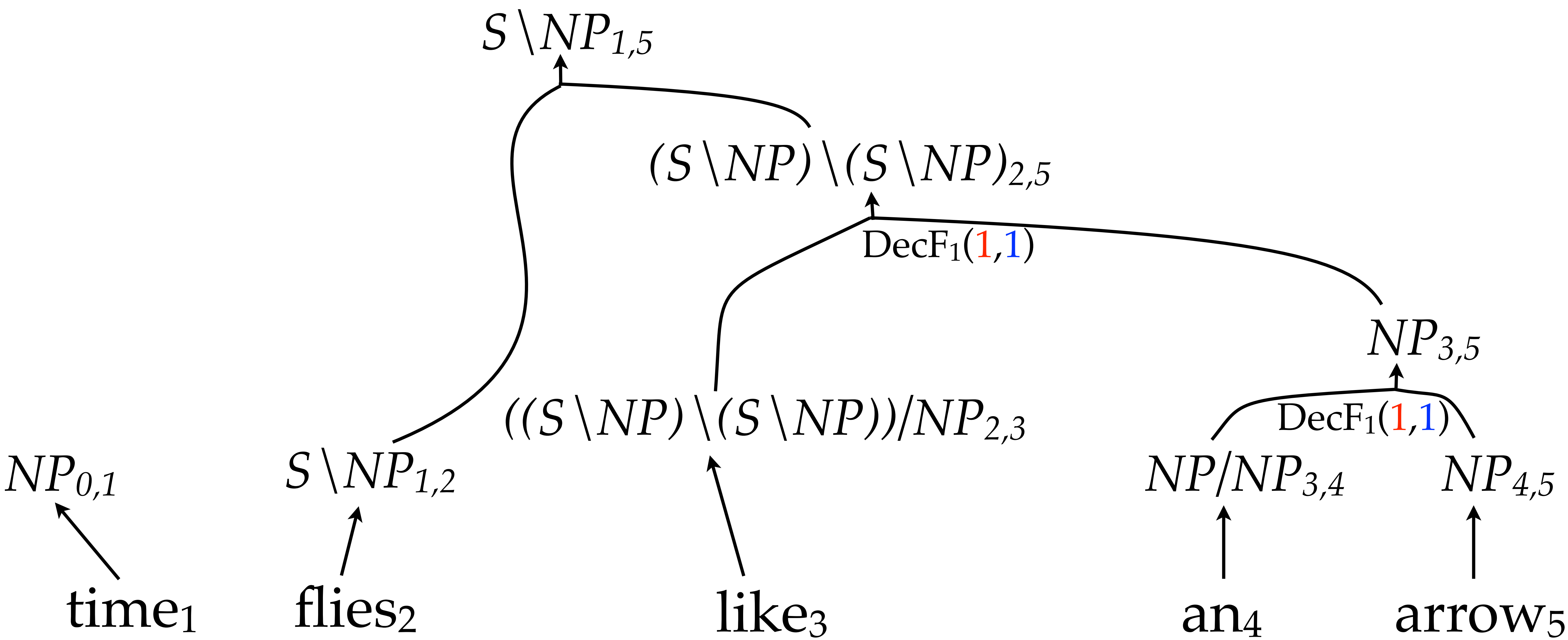
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

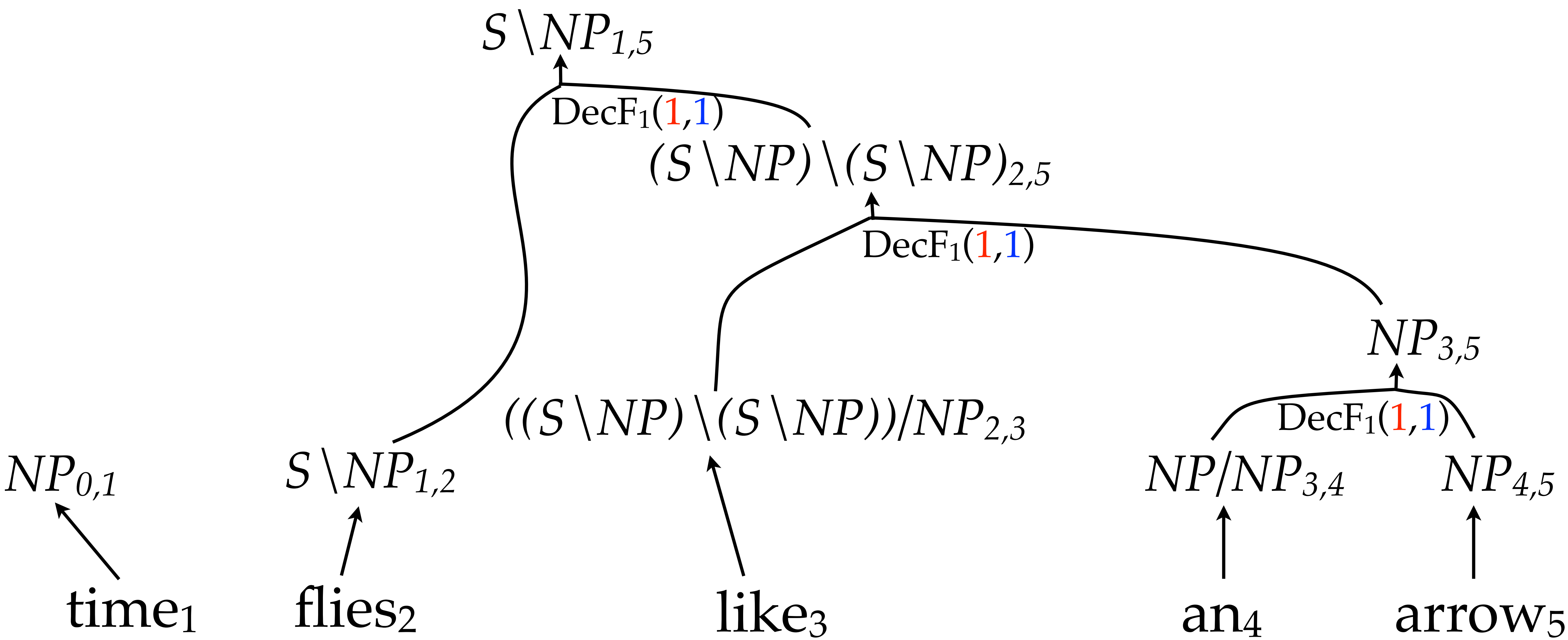
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

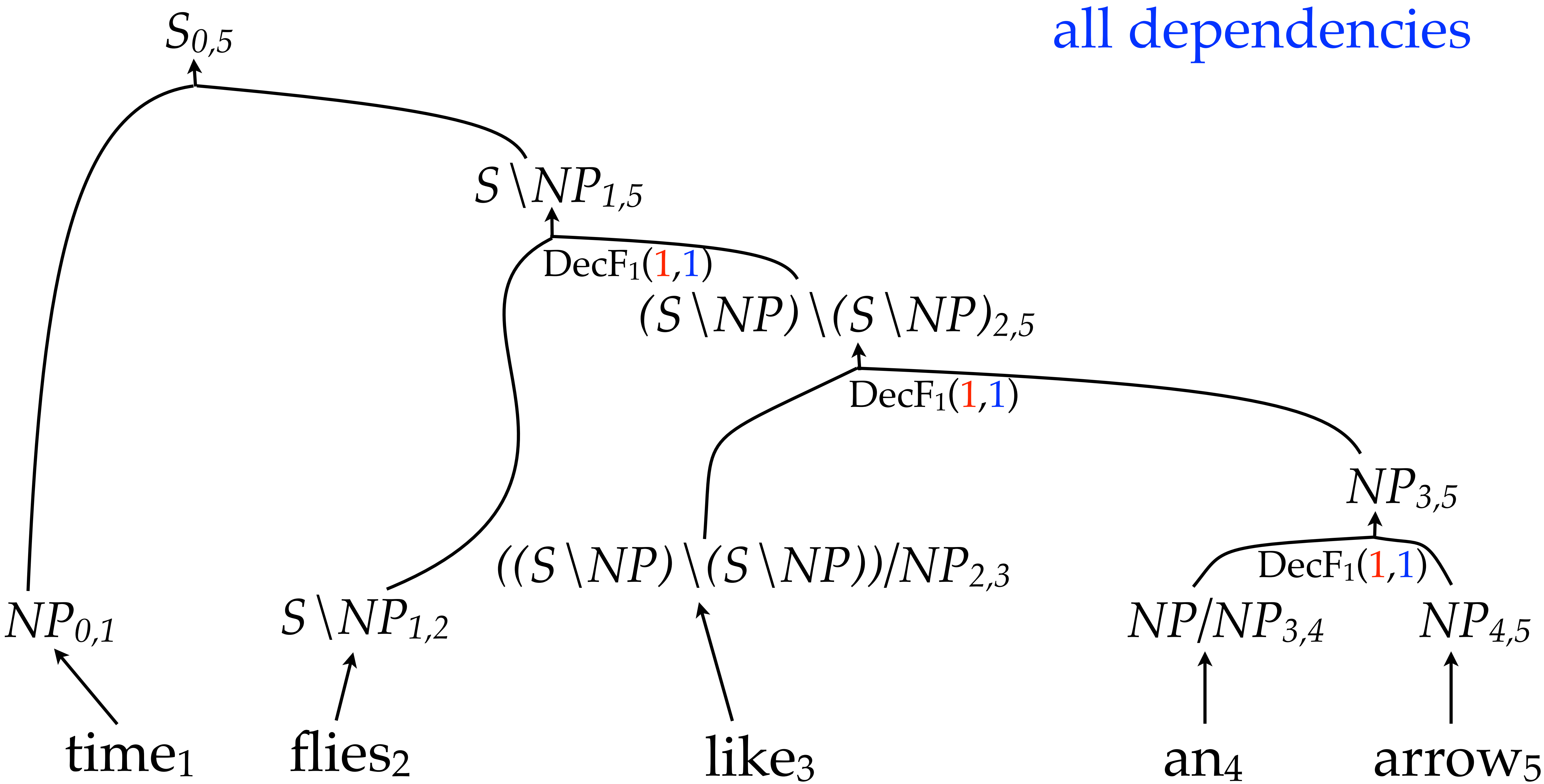
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

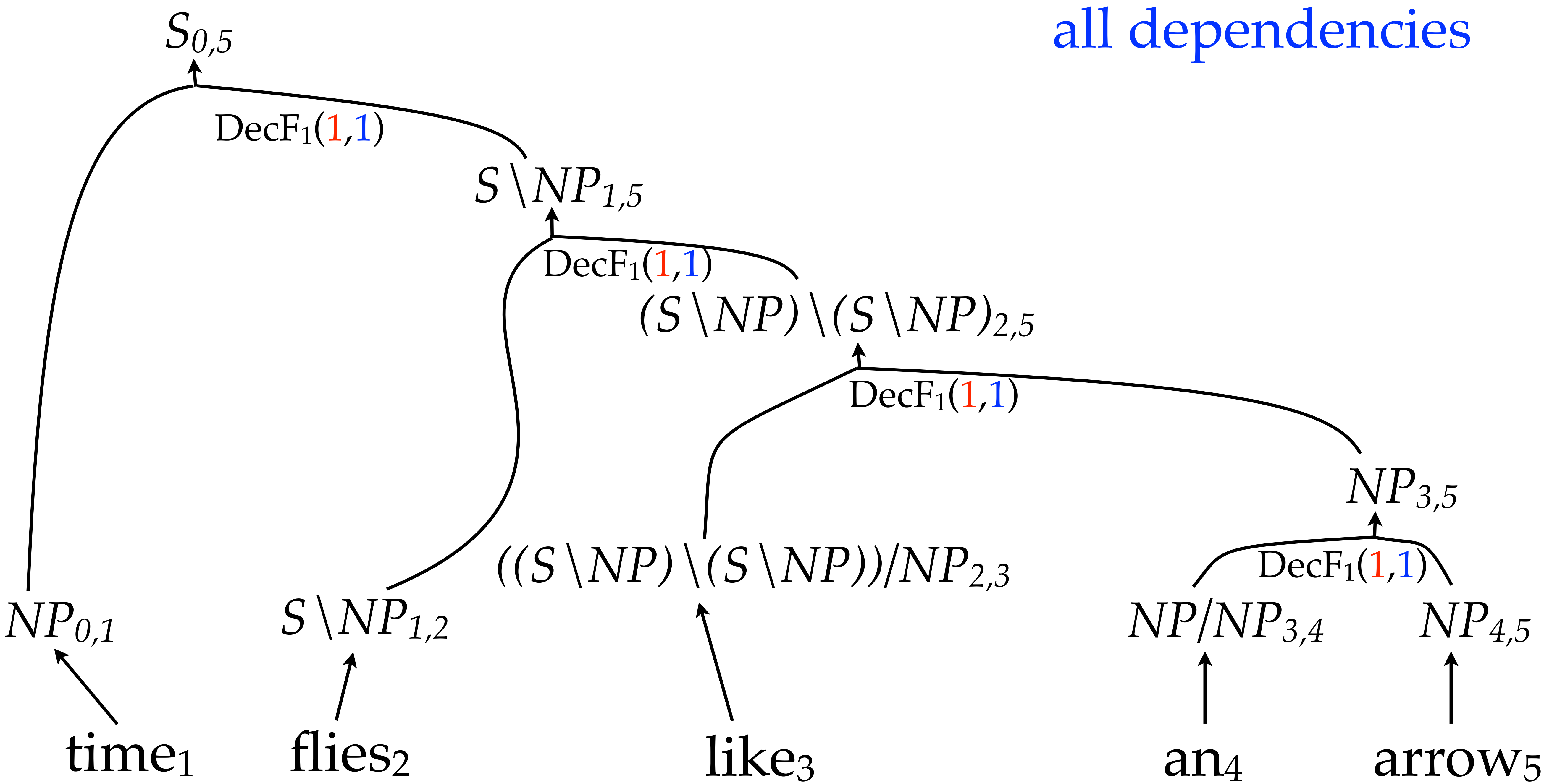
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

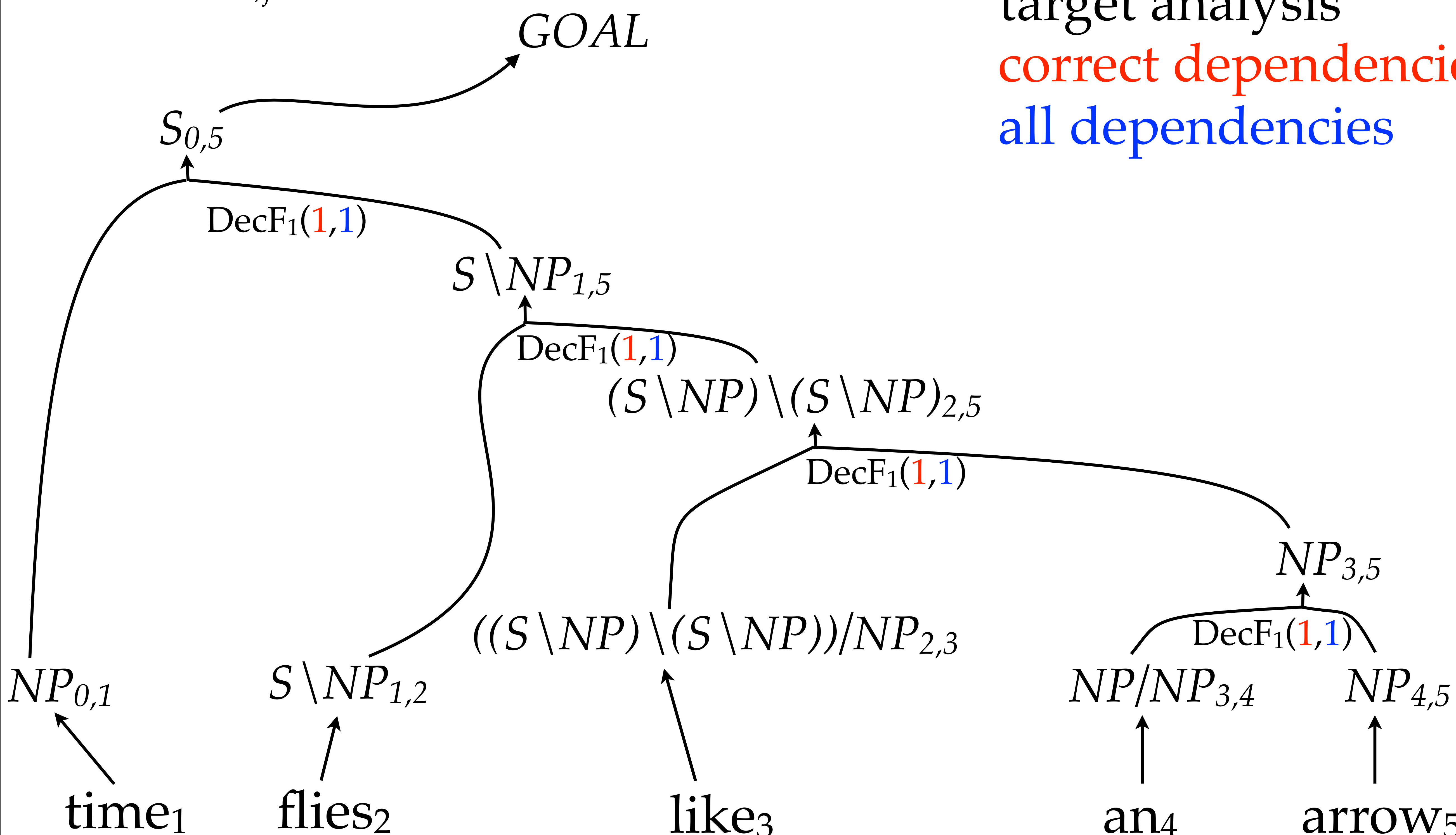
target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

target analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

$(S\backslash NP)/NP_{2,3}$

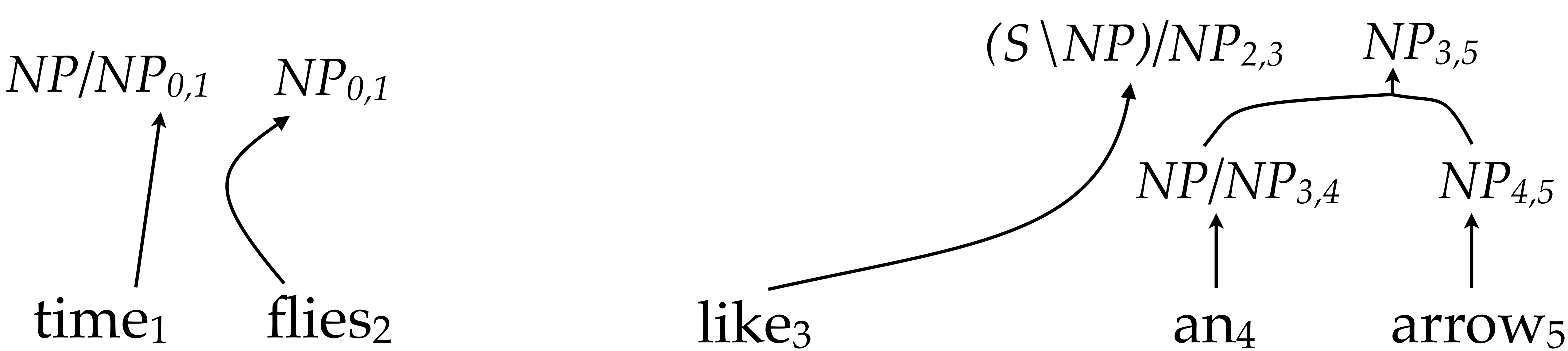
$NP/NP_{0,1}$ $NP_{0,1}$ $NP/NP_{3,4}$ $NP_{4,5}$

time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

$NP/NP_{0,1}$ $\quad NP_{0,1}$ $\quad (S\backslash NP)/NP_{2,3}$ $\quad NP_{3,5}$

$\mathrm{DecF}_1(1,1)$

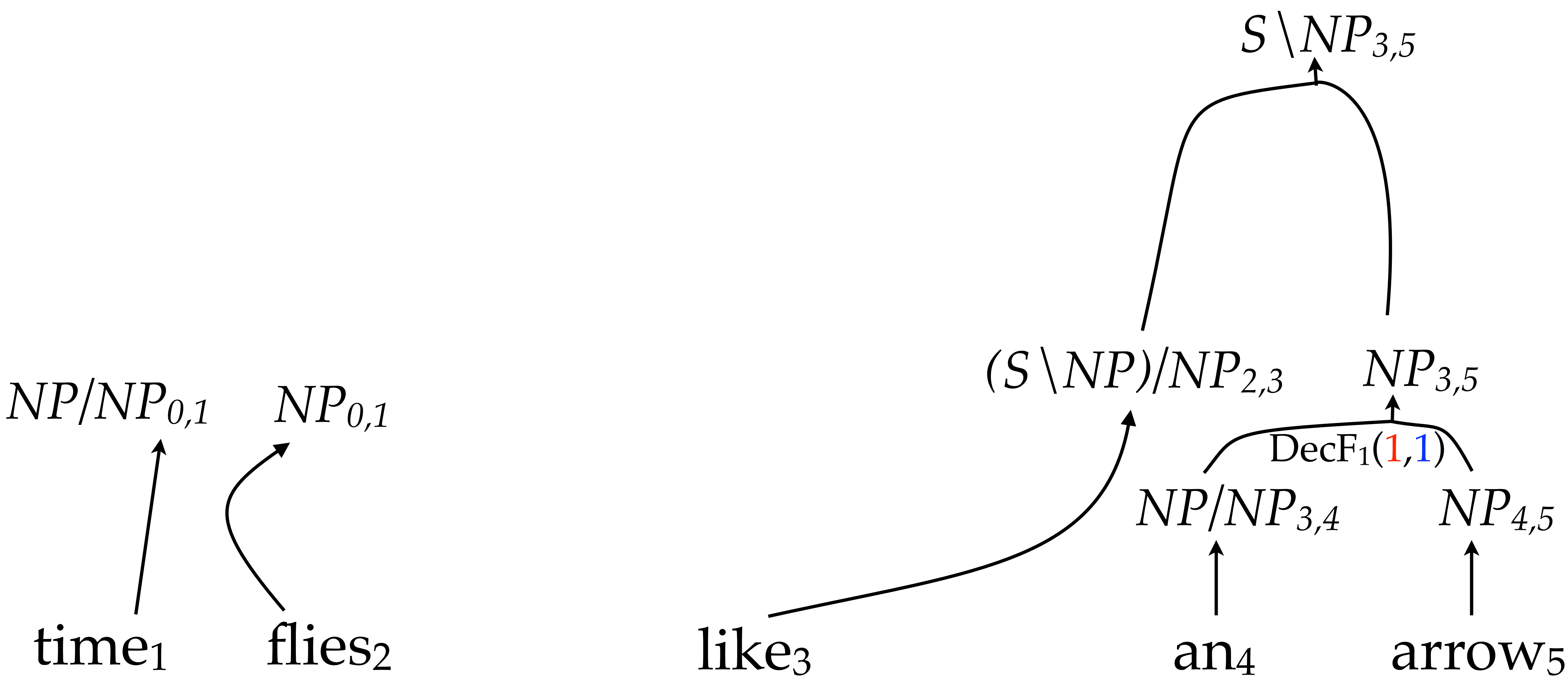
$NP/NP_{3,4}$ $\quad NP_{4,5}$

time_1 $\quad \text{flies}_2$ $\quad \text{like}_3$ $\quad \text{an}_4$ $\quad \text{arrow}_5$

Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies



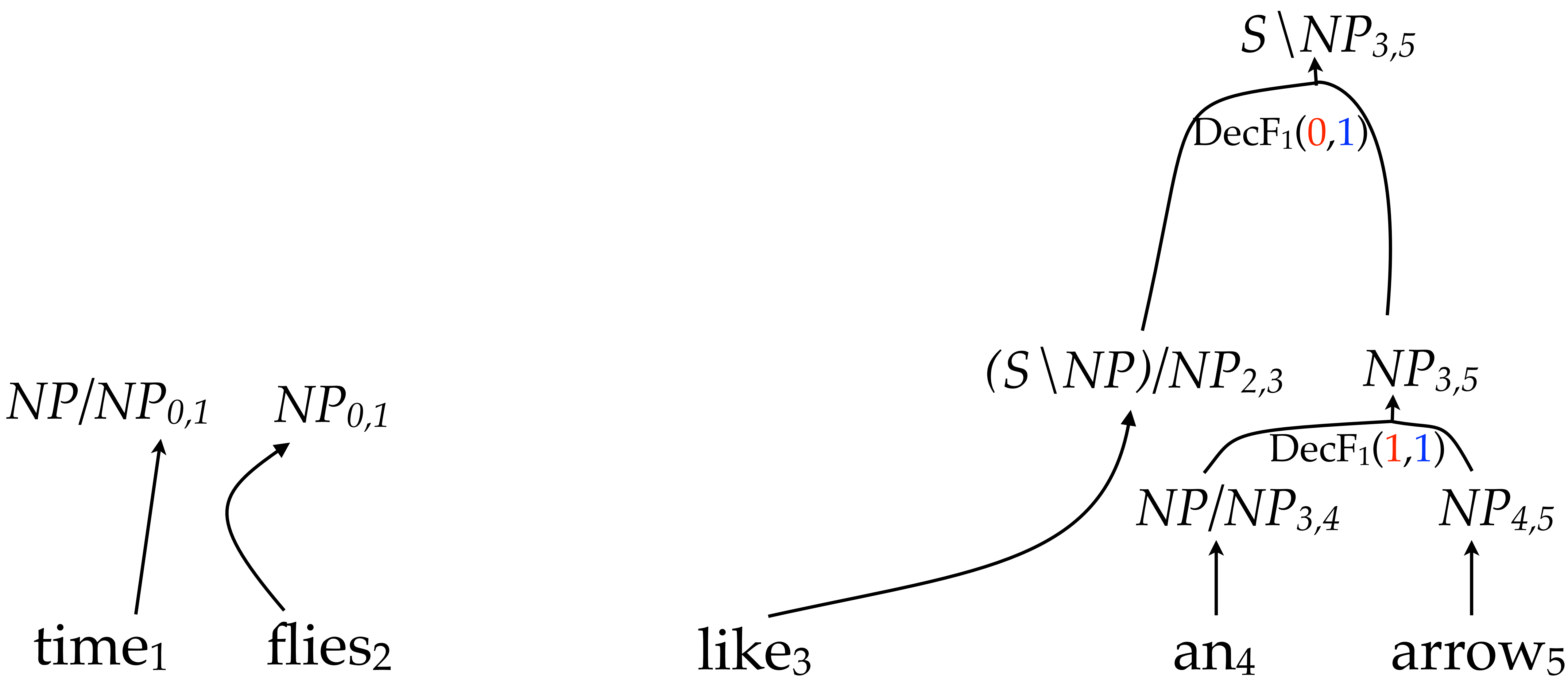
Approximate Losses with CKY

items $A_{i,j}$

another analysis

correct dependencies

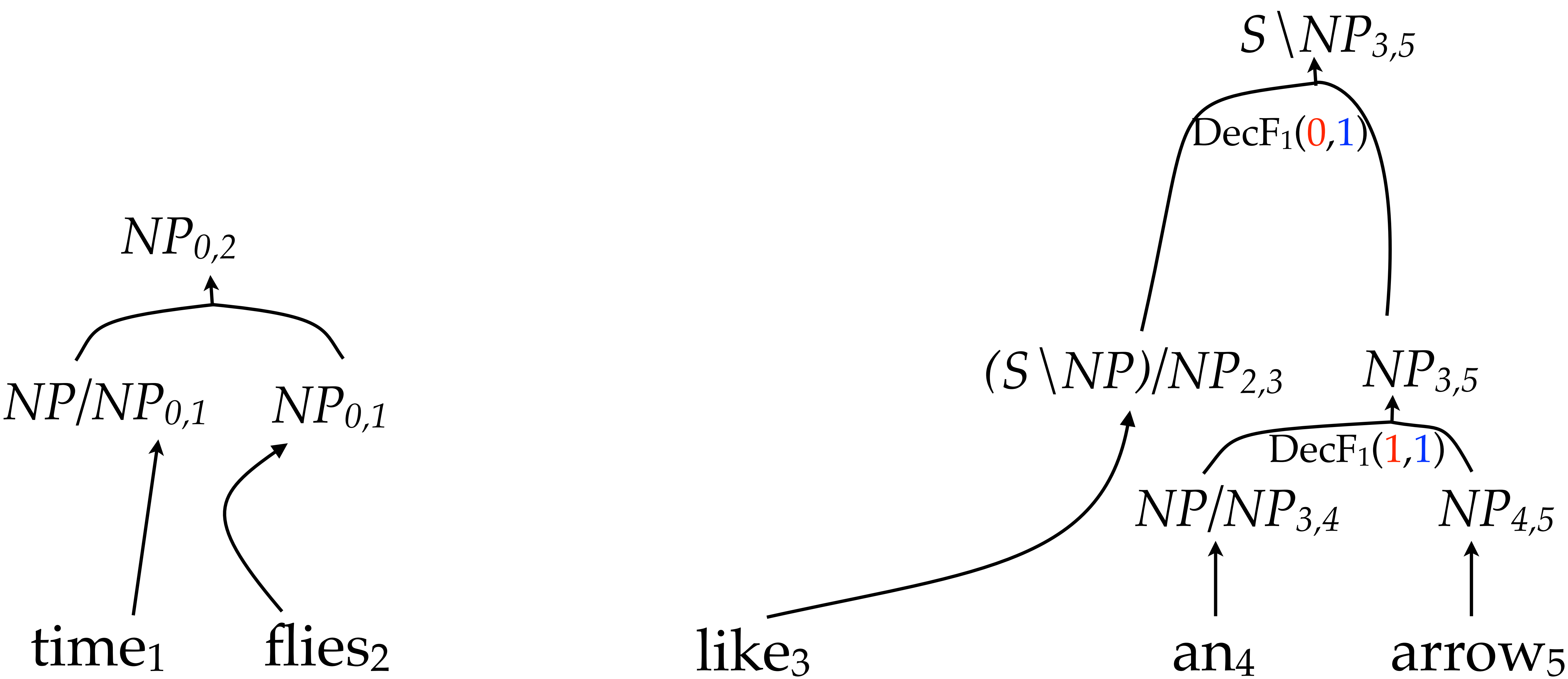
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies



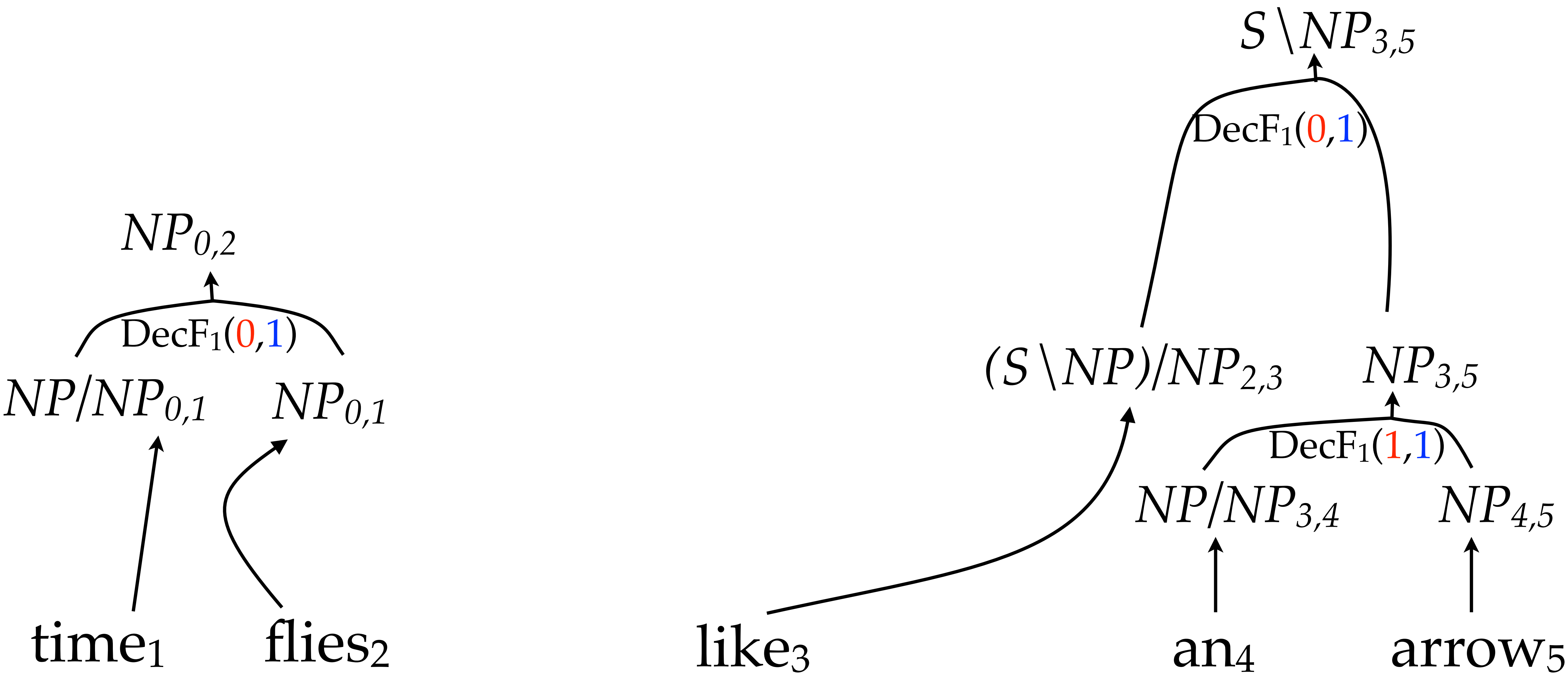
Approximate Losses with CKY

items $A_{i,j}$

another analysis

correct dependencies

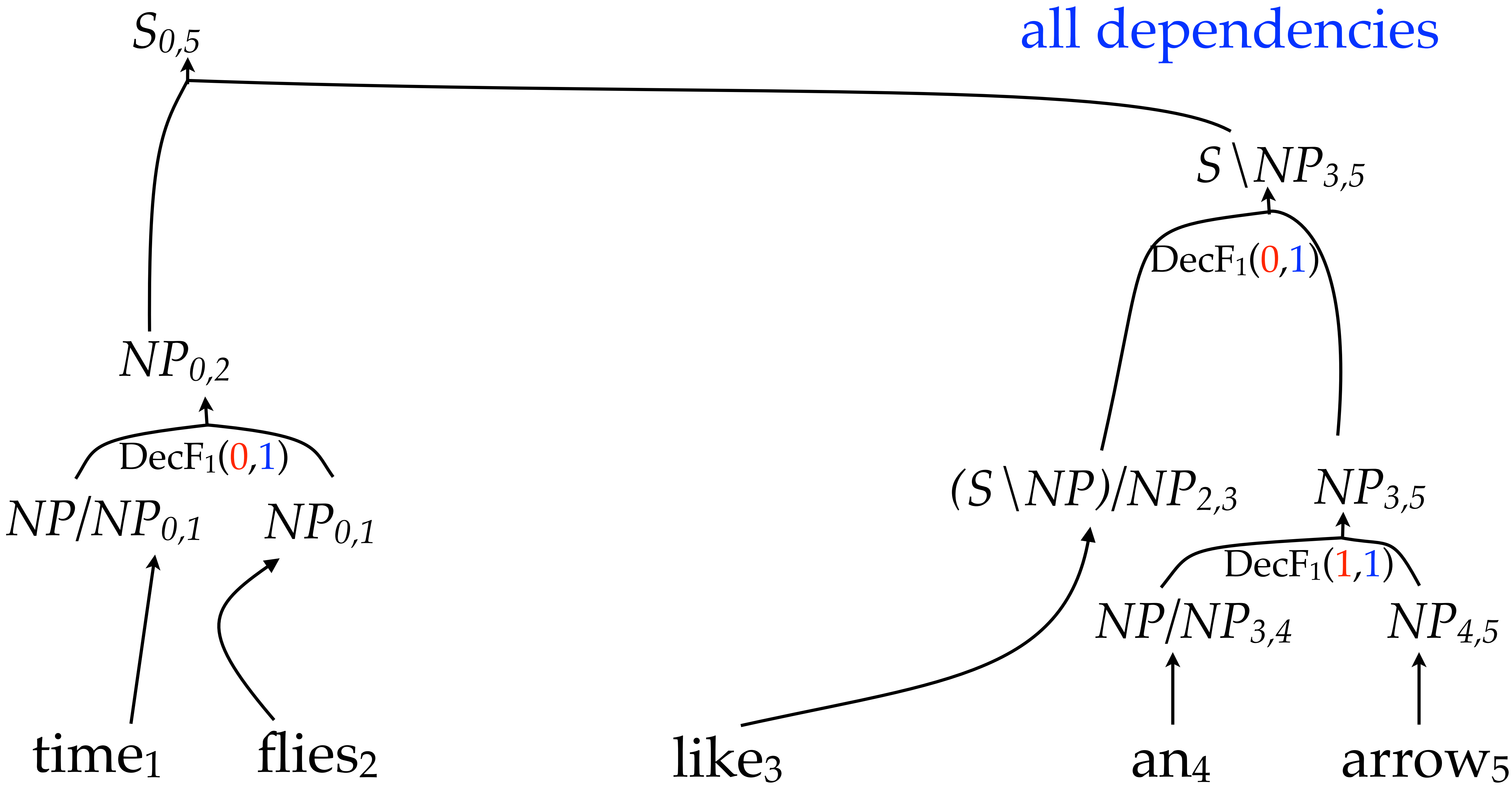
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

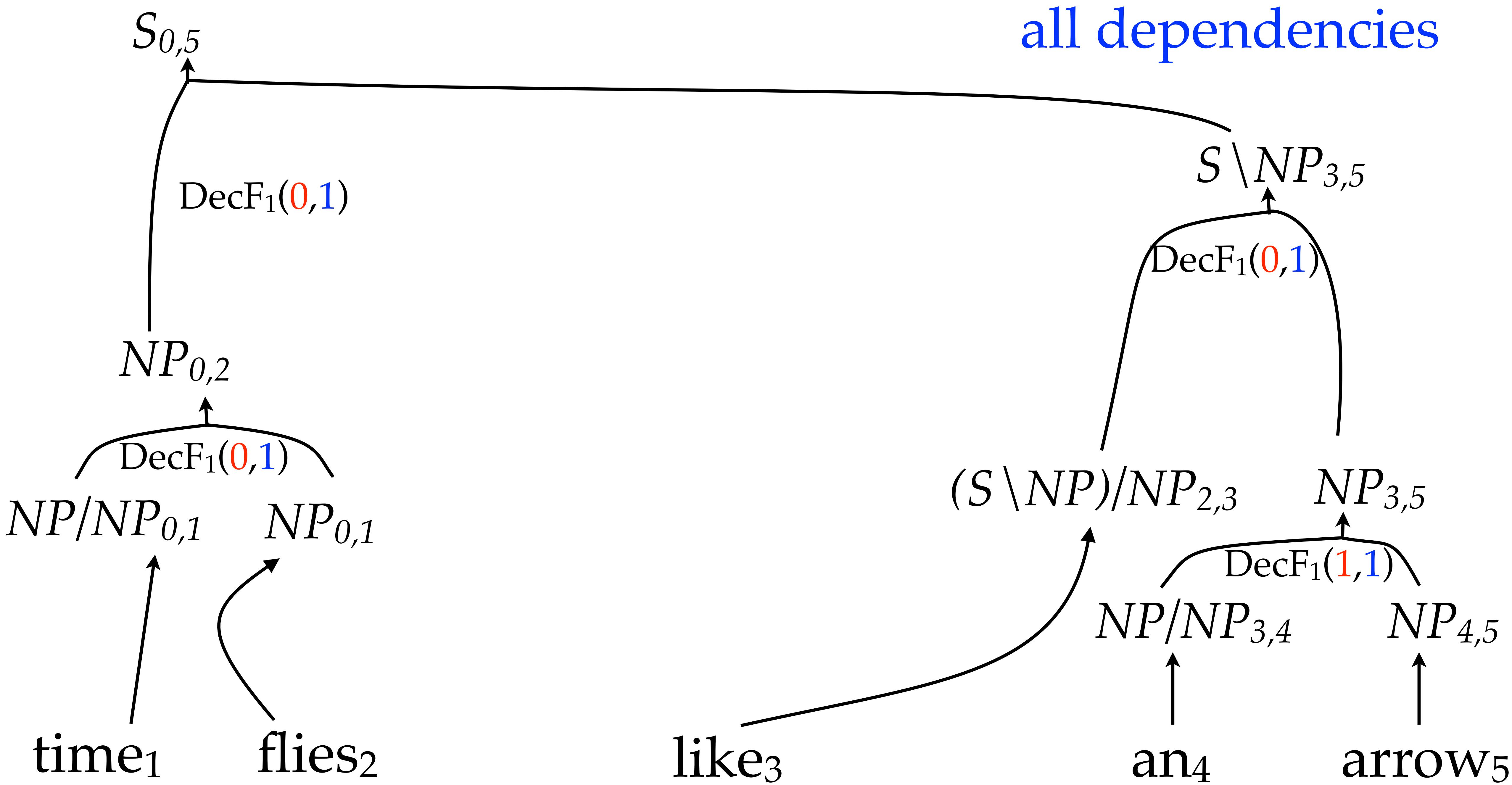
another analysis
correct dependencies
all dependencies



Approximate Losses with CKY

items $A_{i,j}$

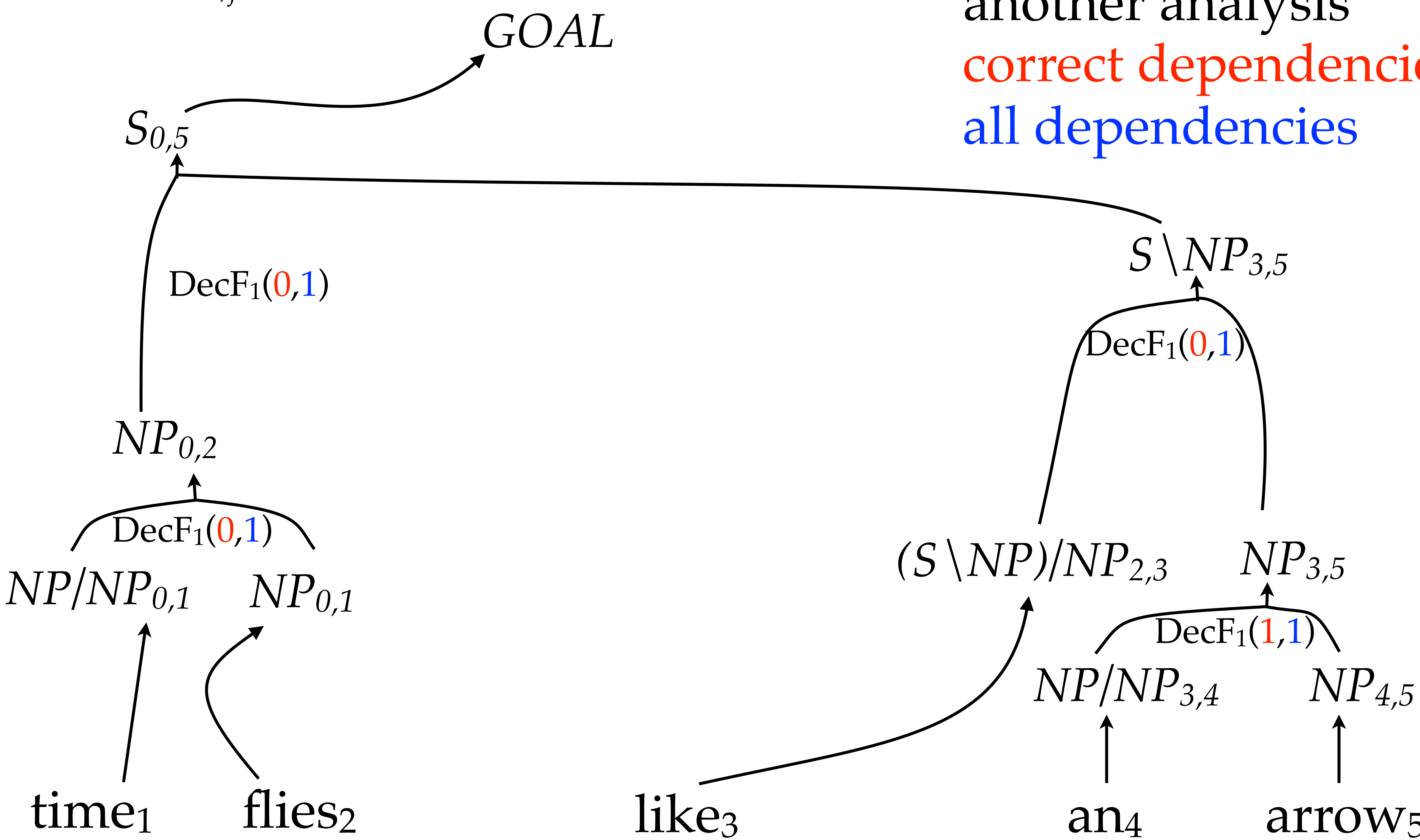
another analysis
correct dependencies
all dependencies



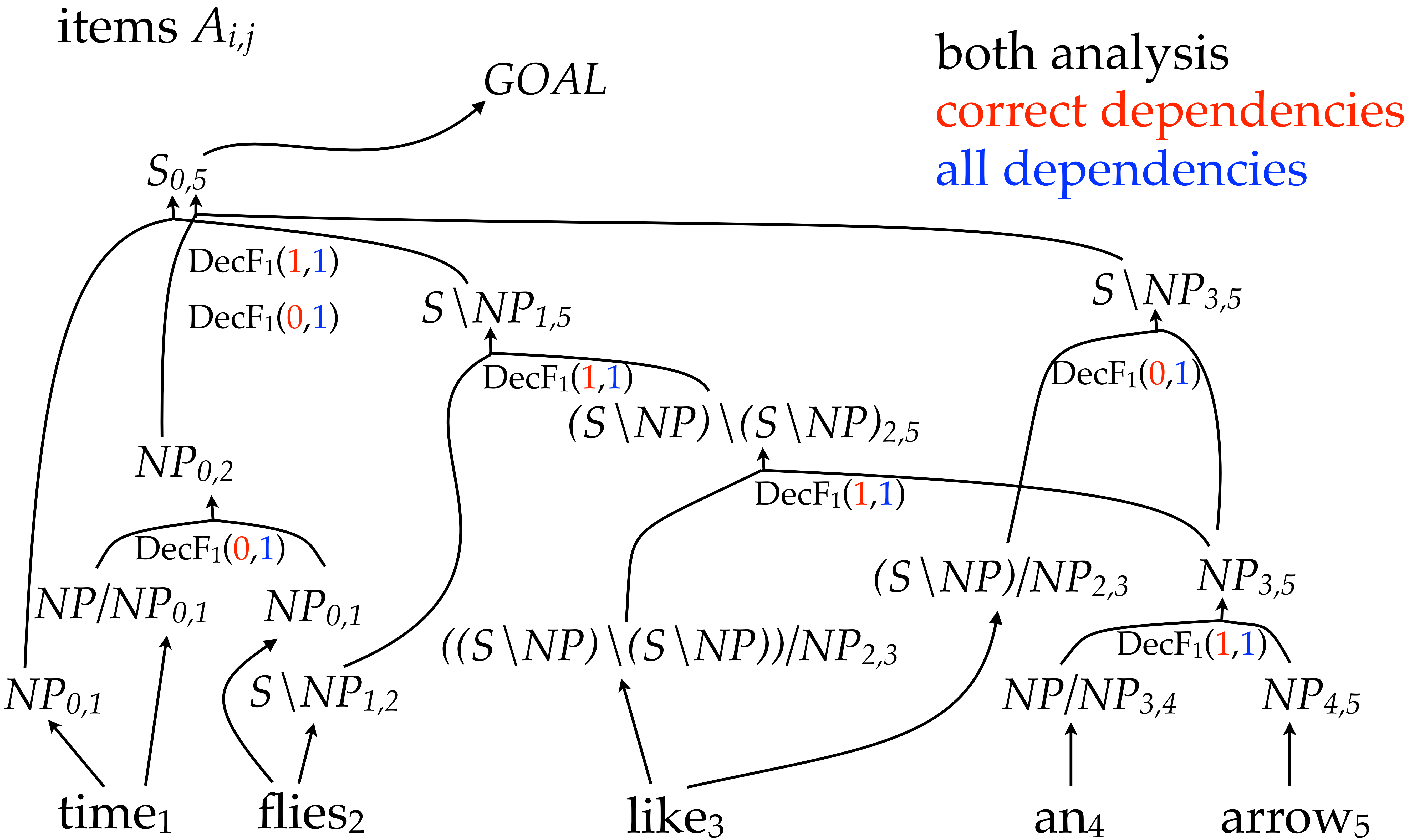
Approximate Losses with CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies



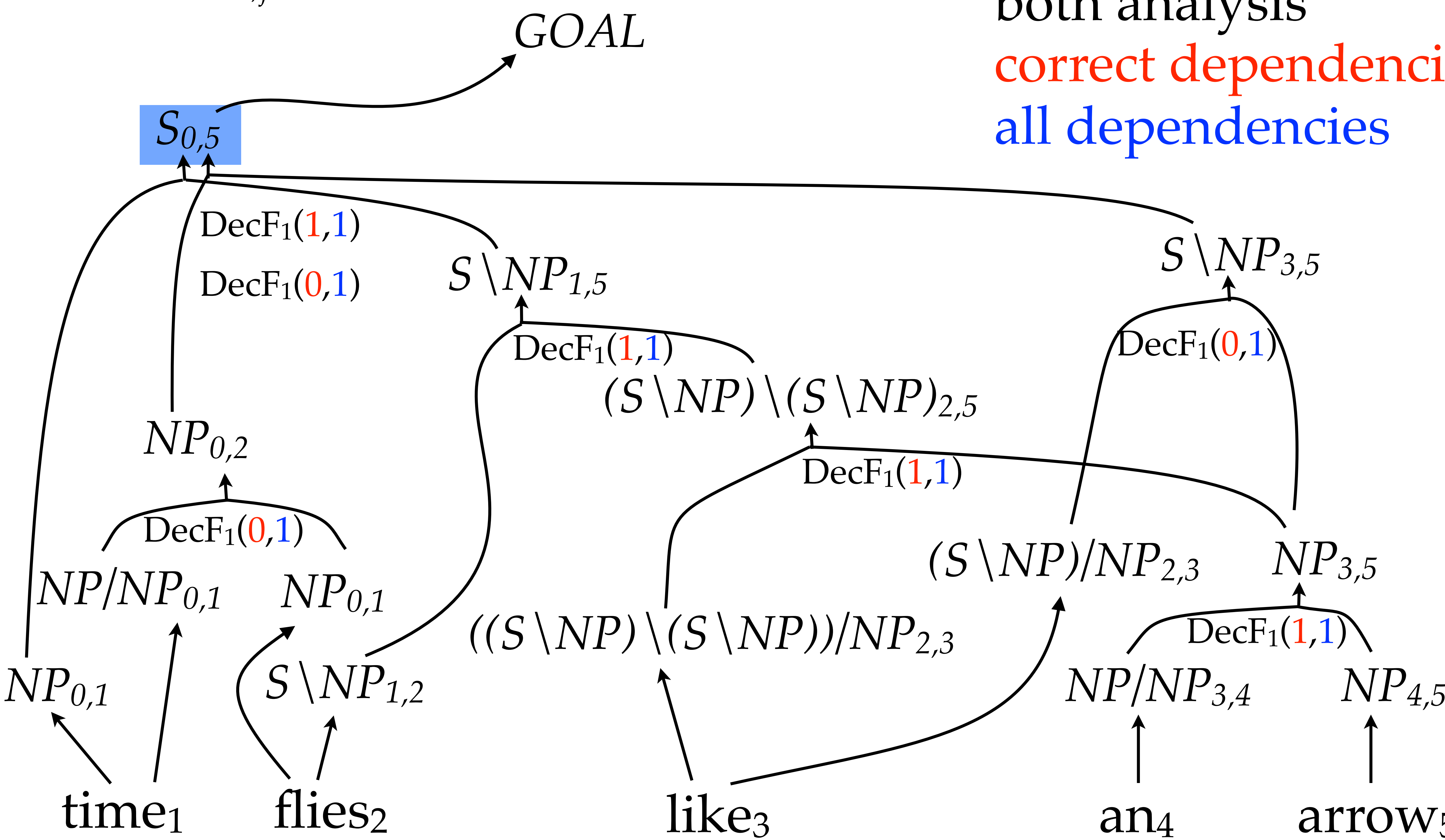
Approximate Losses with CKY



Approximate Losses with CKY

items $A_{i,j}$

both analysis
correct dependencies
all dependencies



Exact Losses for Parsing

Precision $$P(y, y') = \frac{|y \cap y'|}{|y'|} = \frac{n}{d}$$

Recall $$R(y, y') = \frac{|y \cap y'|}{|y|} = \frac{n}{|y|}$$

F-measure $$F_1(y, y') = \frac{2PR}{P + R} = \frac{2|y \cap y'|}{|y| + |y'|} = \frac{2n}{d + |y|}$$

- Compute exact losses on sentence-level i.e. items $A_{i,j,n,d}$
- Treat F_1 as *non-local features* dependent on n, d

Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

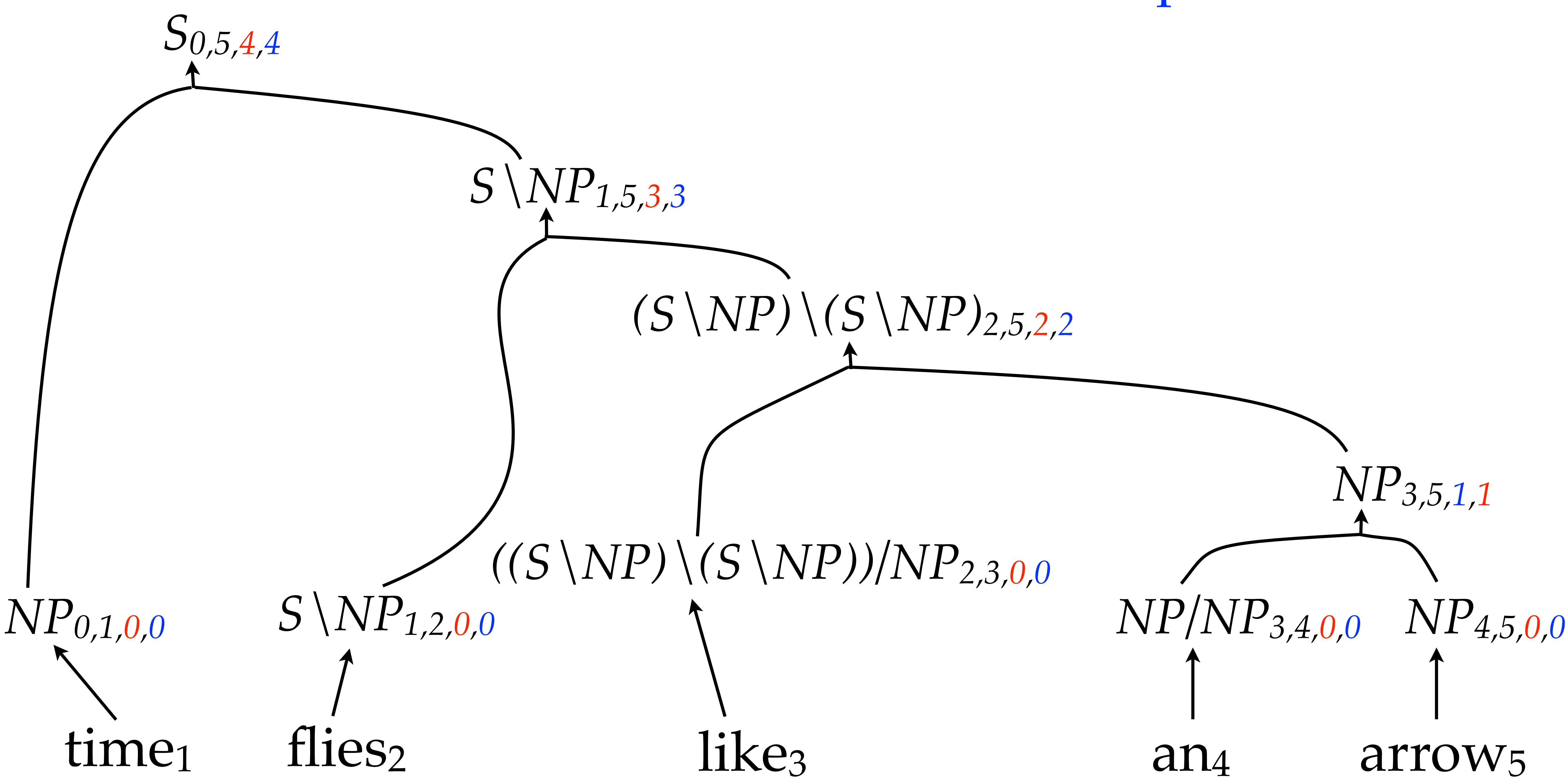
time_1 flies_2 like_3 an_4 arrow_5

Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

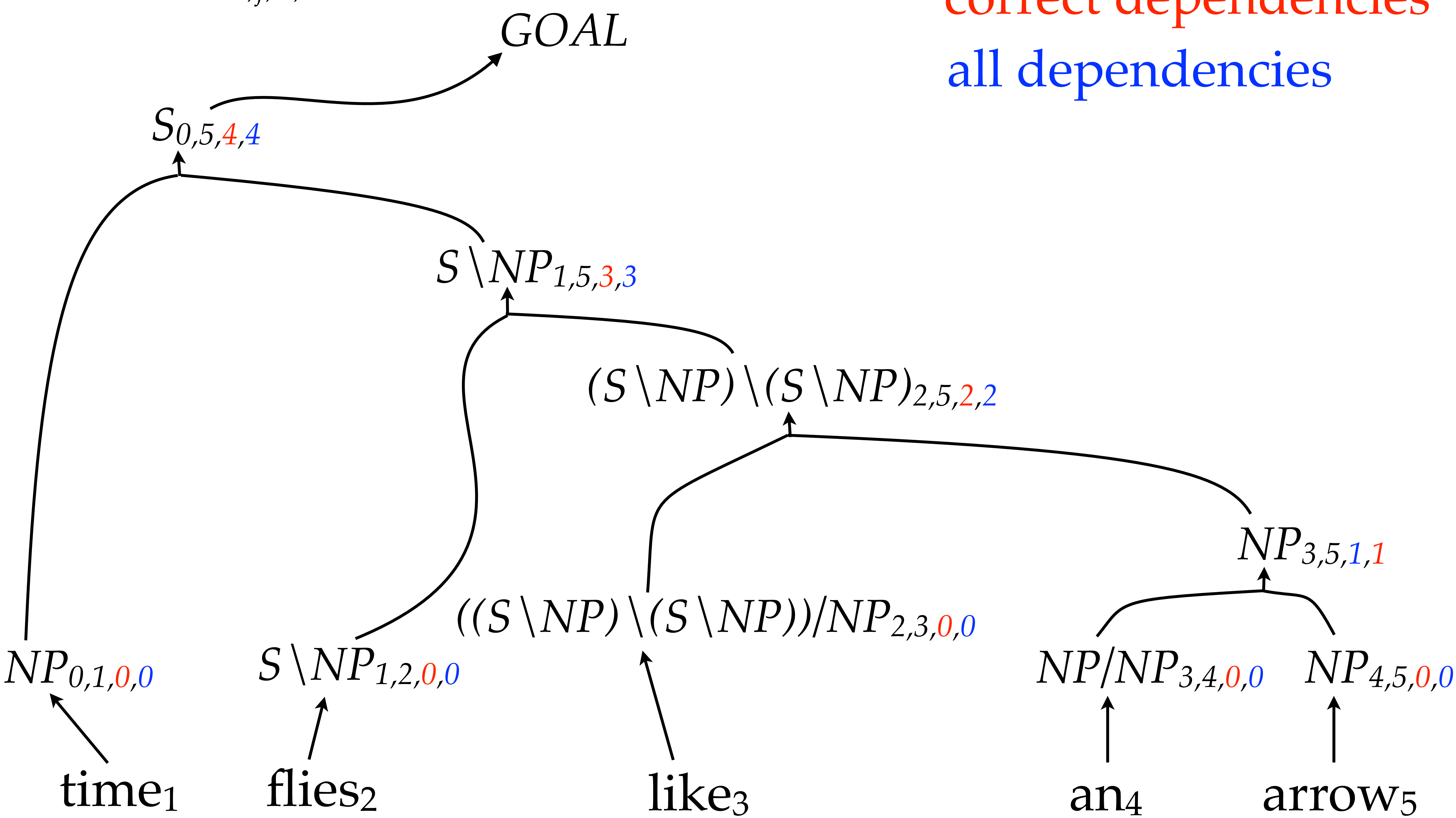


Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

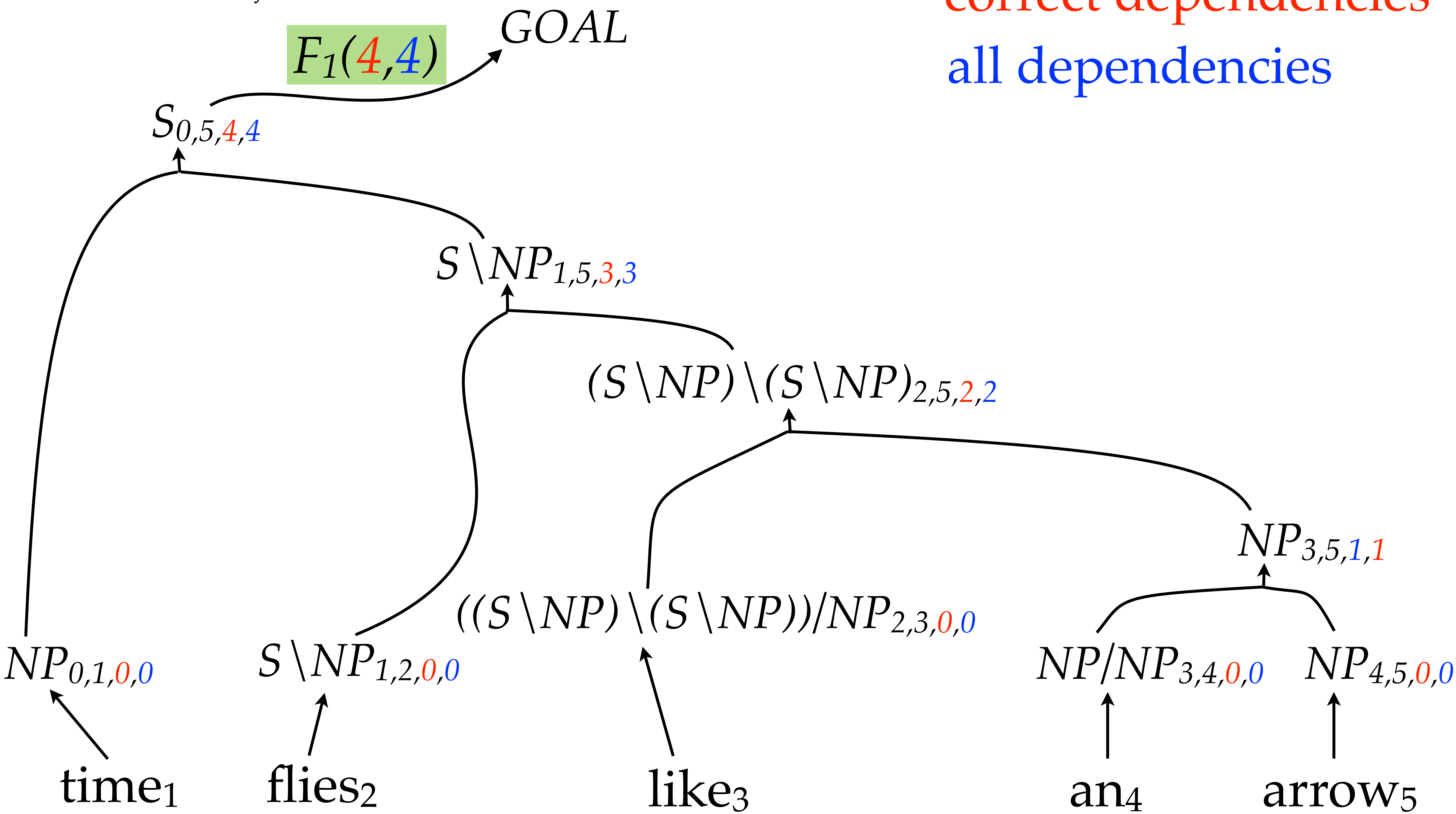


Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies



Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

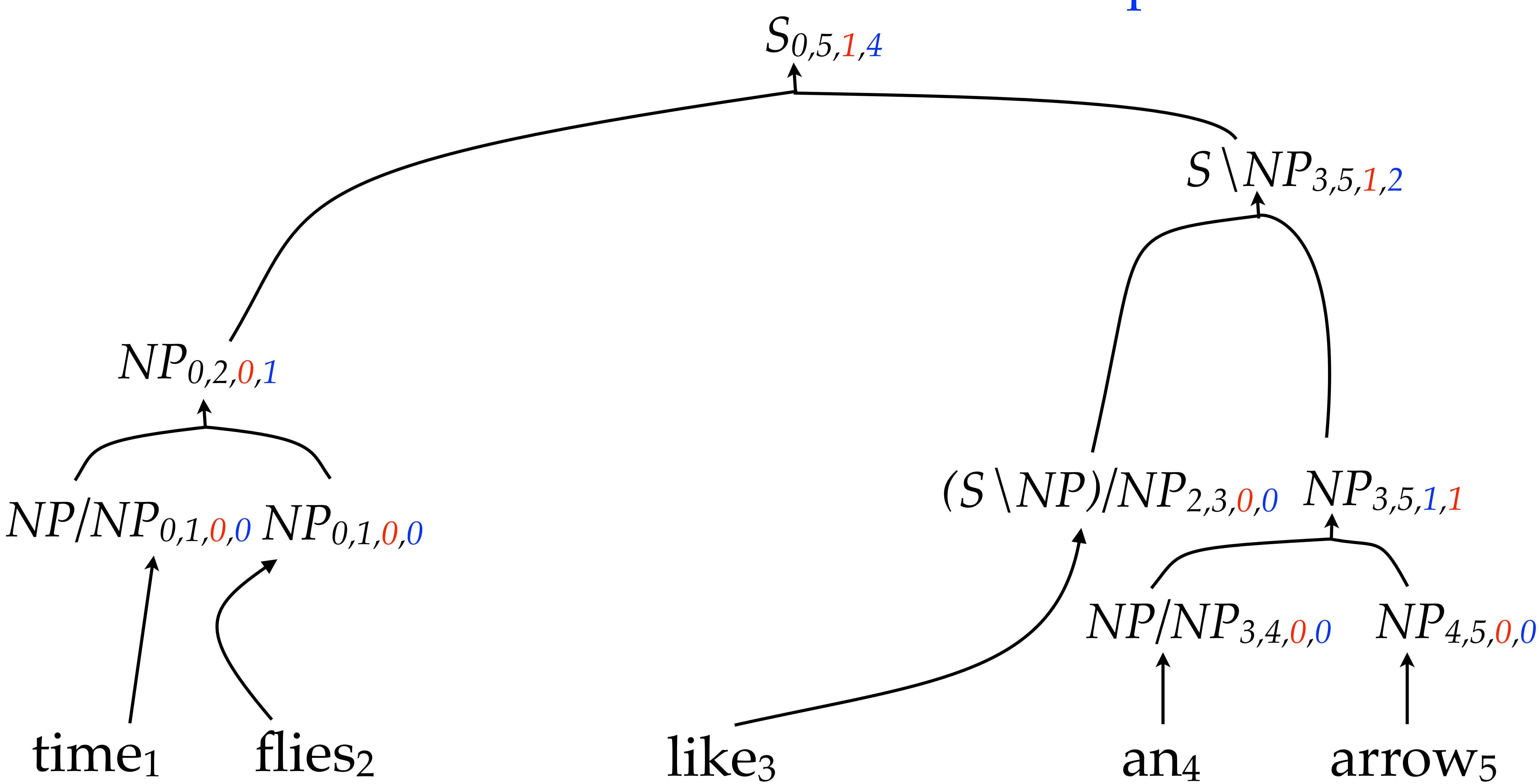
time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

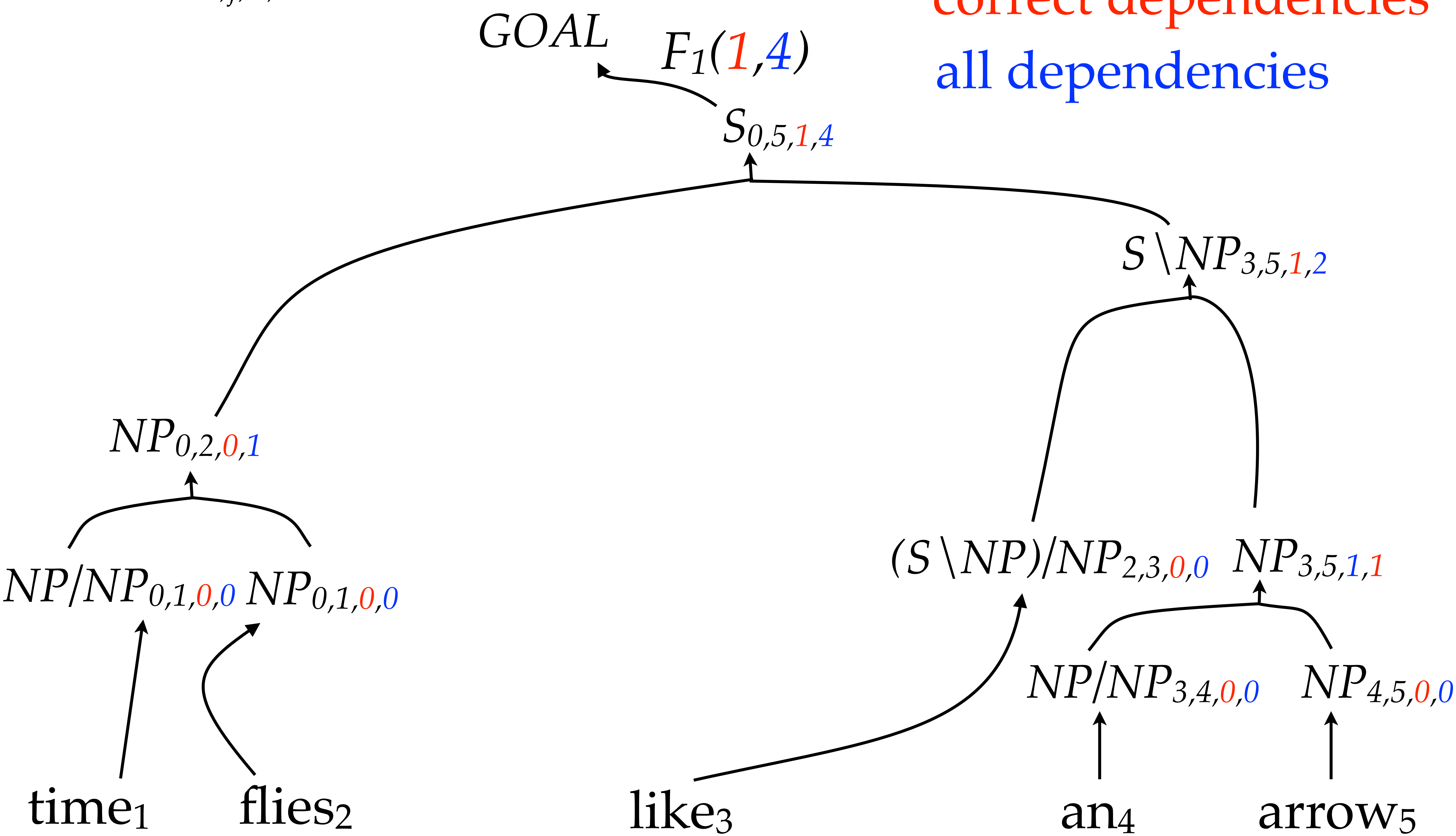


Exact Losses with State-Split CKY

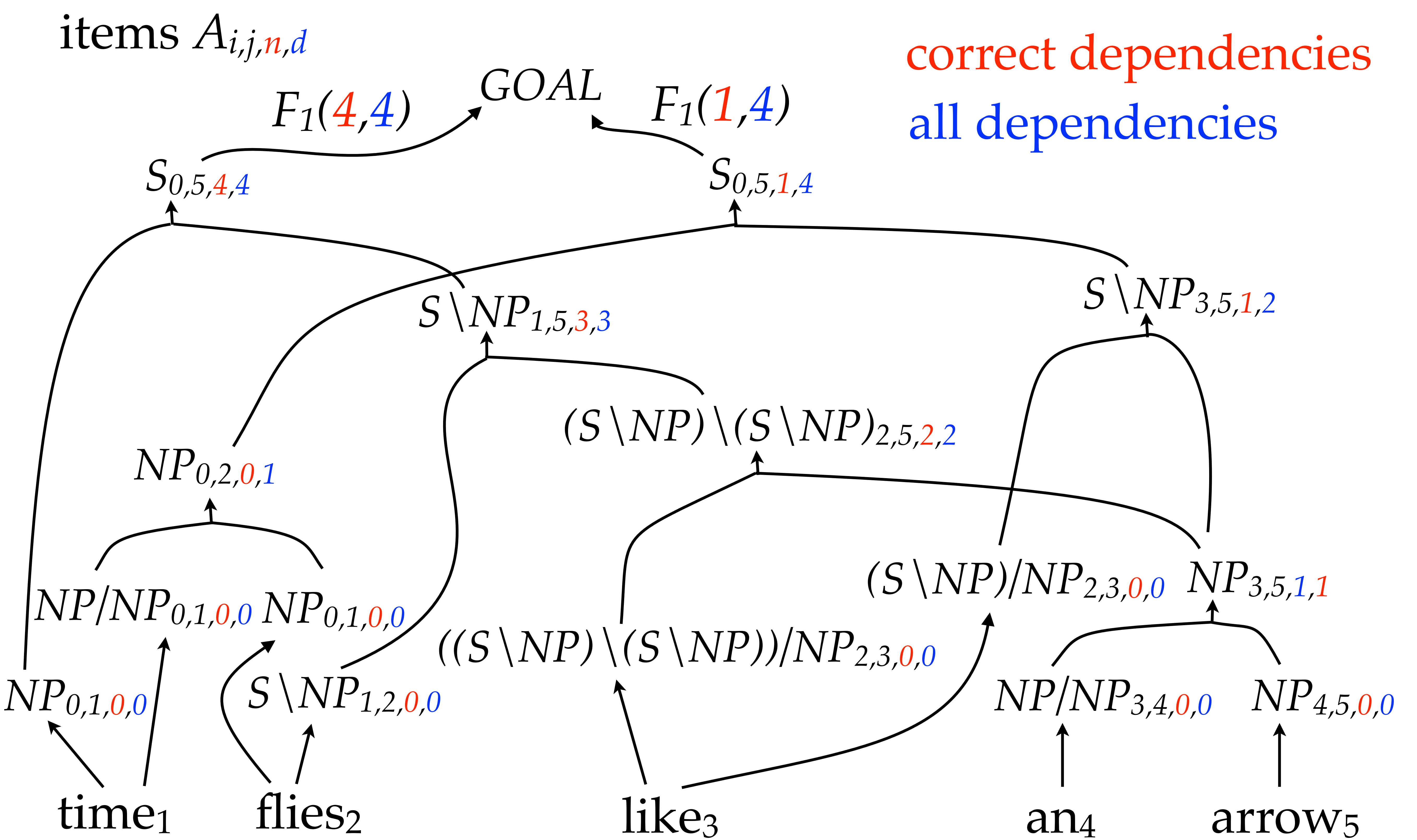
items $A_{i,j,n,d}$

correct dependencies

all dependencies



Exact Losses with State-Split CKY

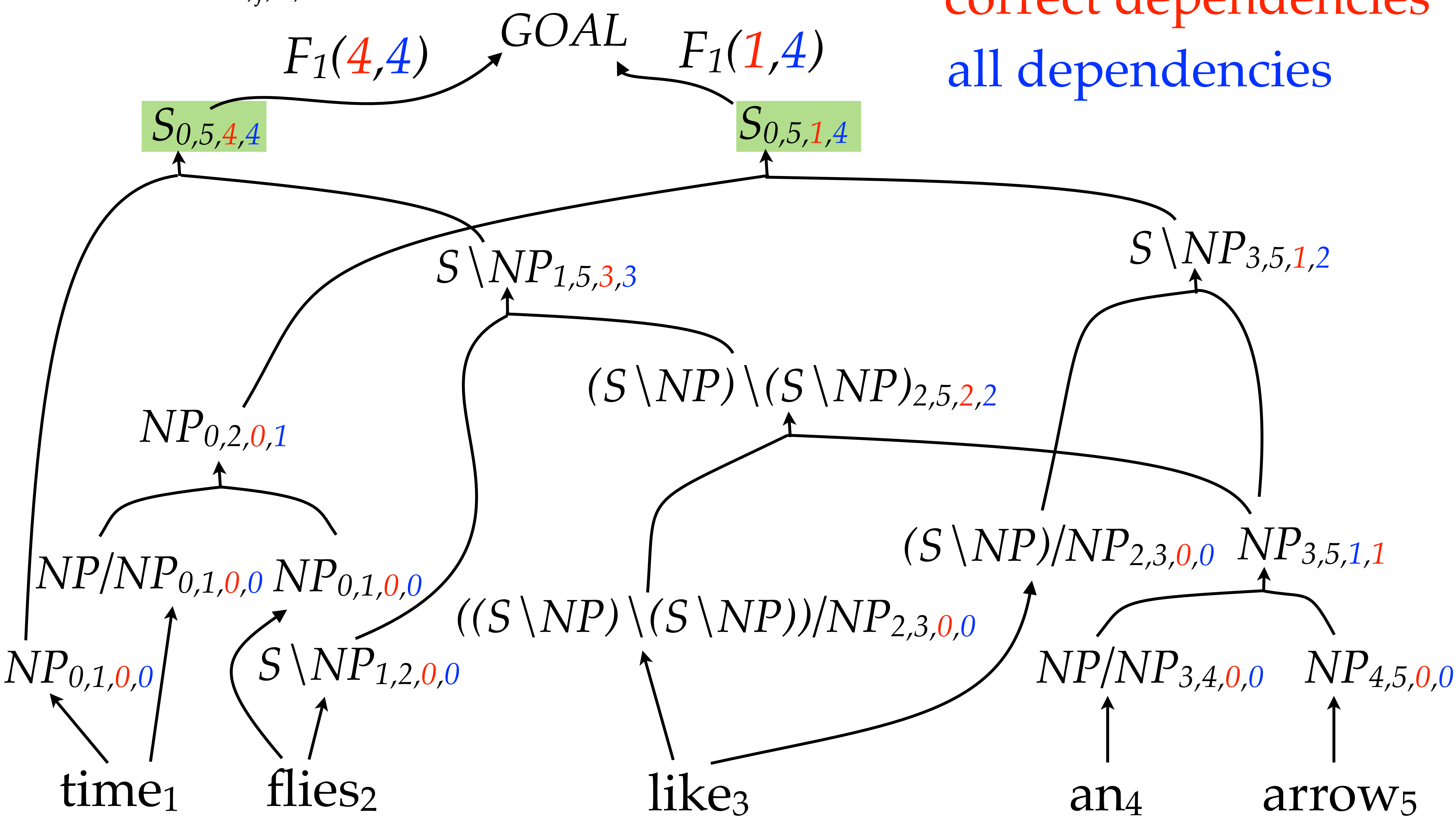


Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

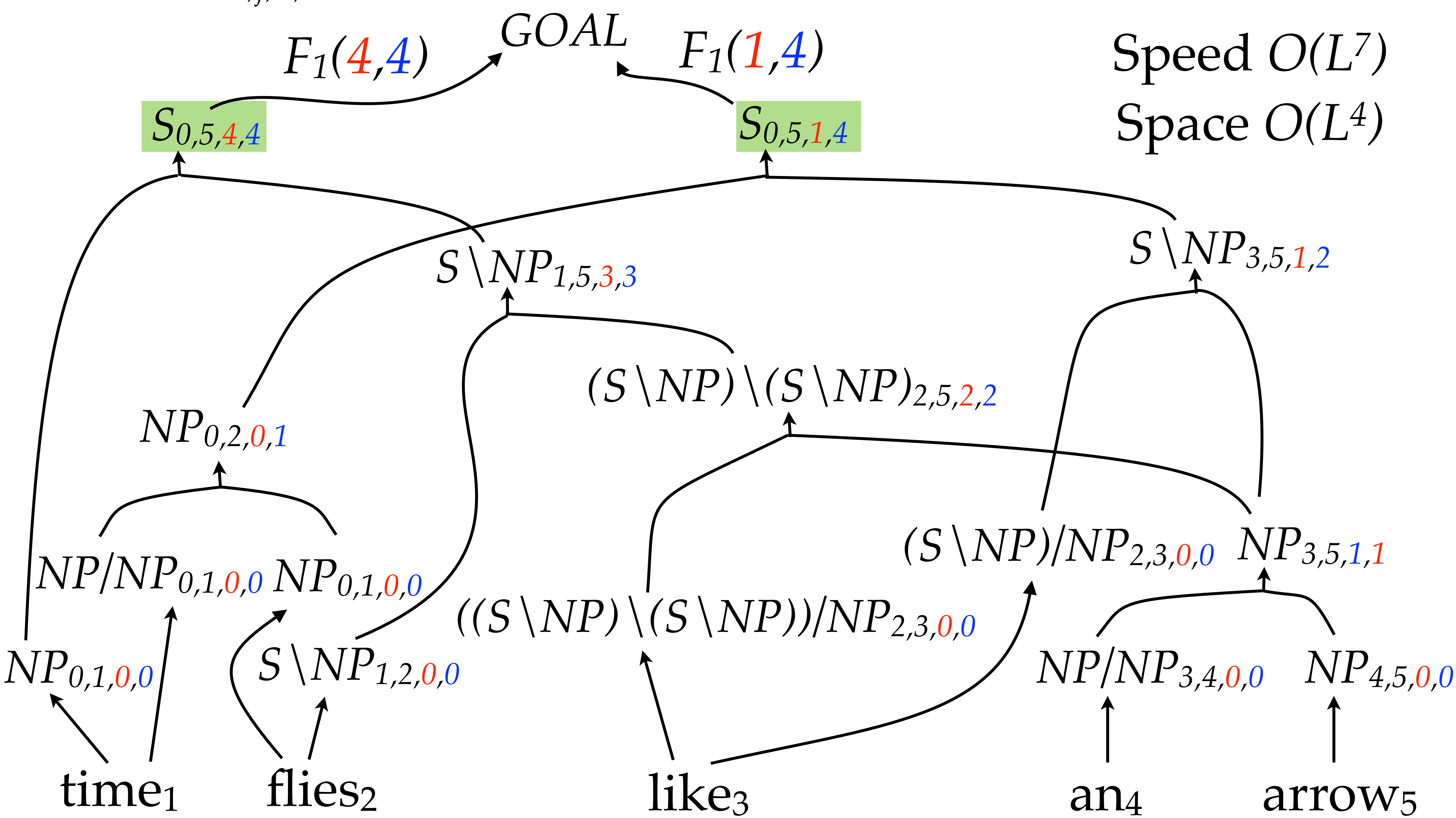


Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

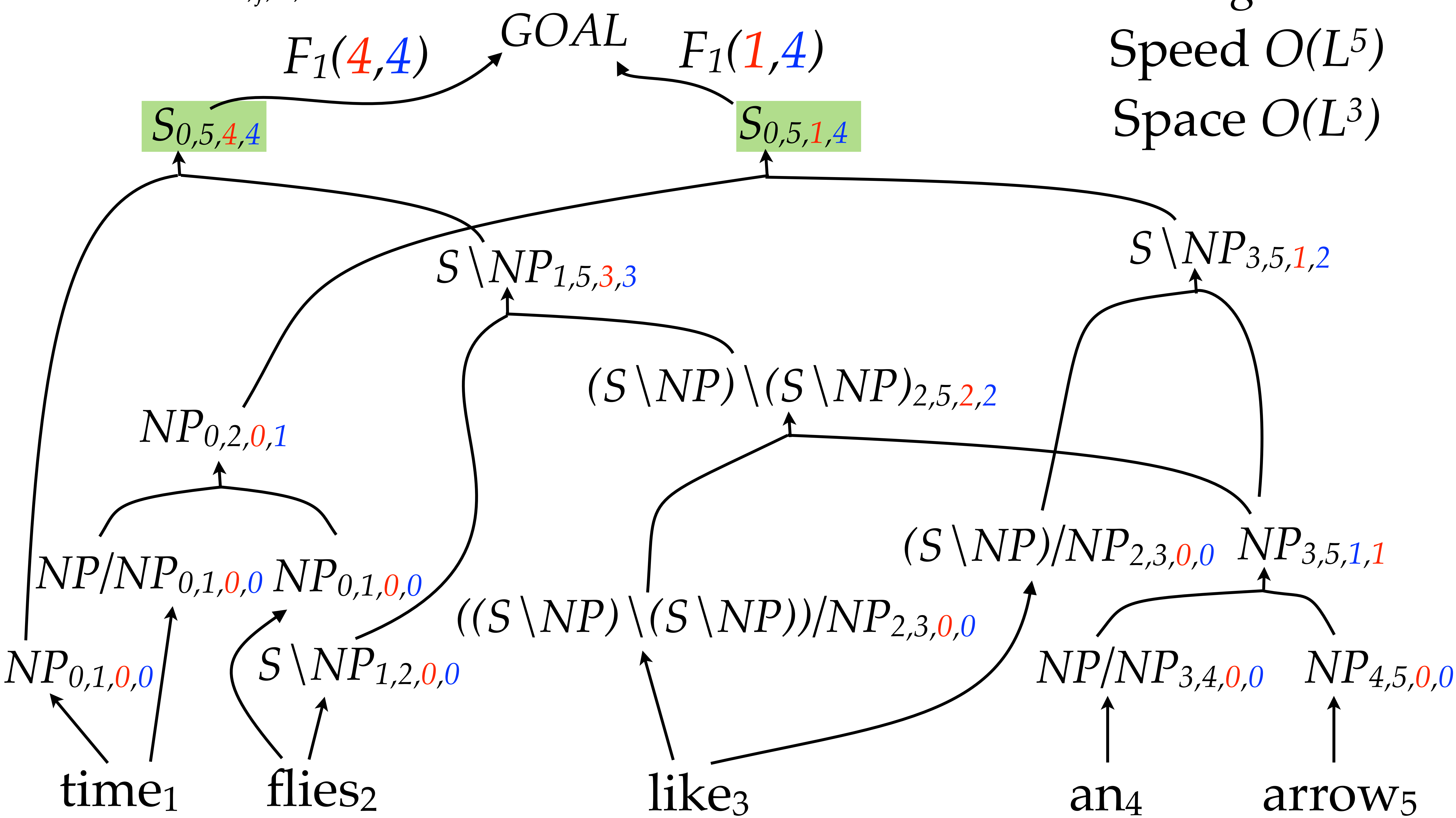
Speed $O(L^7)$

Space $O(L^4)$



Exact Losses with State-Split CKY

items $A_{i,j,n,d}$



average case

Speed $O(L^5)$

Space $O(L^3)$

Experiments: Baseline + SMM

- Exact versus approximate loss functions on test:

Loss	Approx	Exact

Experiments: Baseline + SMM

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23

Experiments: Baseline + SMM

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23
Recall	87.42	87.51

Experiments: Baseline + SMM

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23
Recall	87.42	87.51
F_1	87.69	87.71

Experiments: Baseline + SMM

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23
Recall	87.42	87.51
F_1	87.69	87.71

Approximate loss functions very competitive!

Experiments: Baseline + SMM

- Results on test:
- CLL
  - tight beam: 87.73
  - loose beam: 87.65

Experiments: Baseline + SMM

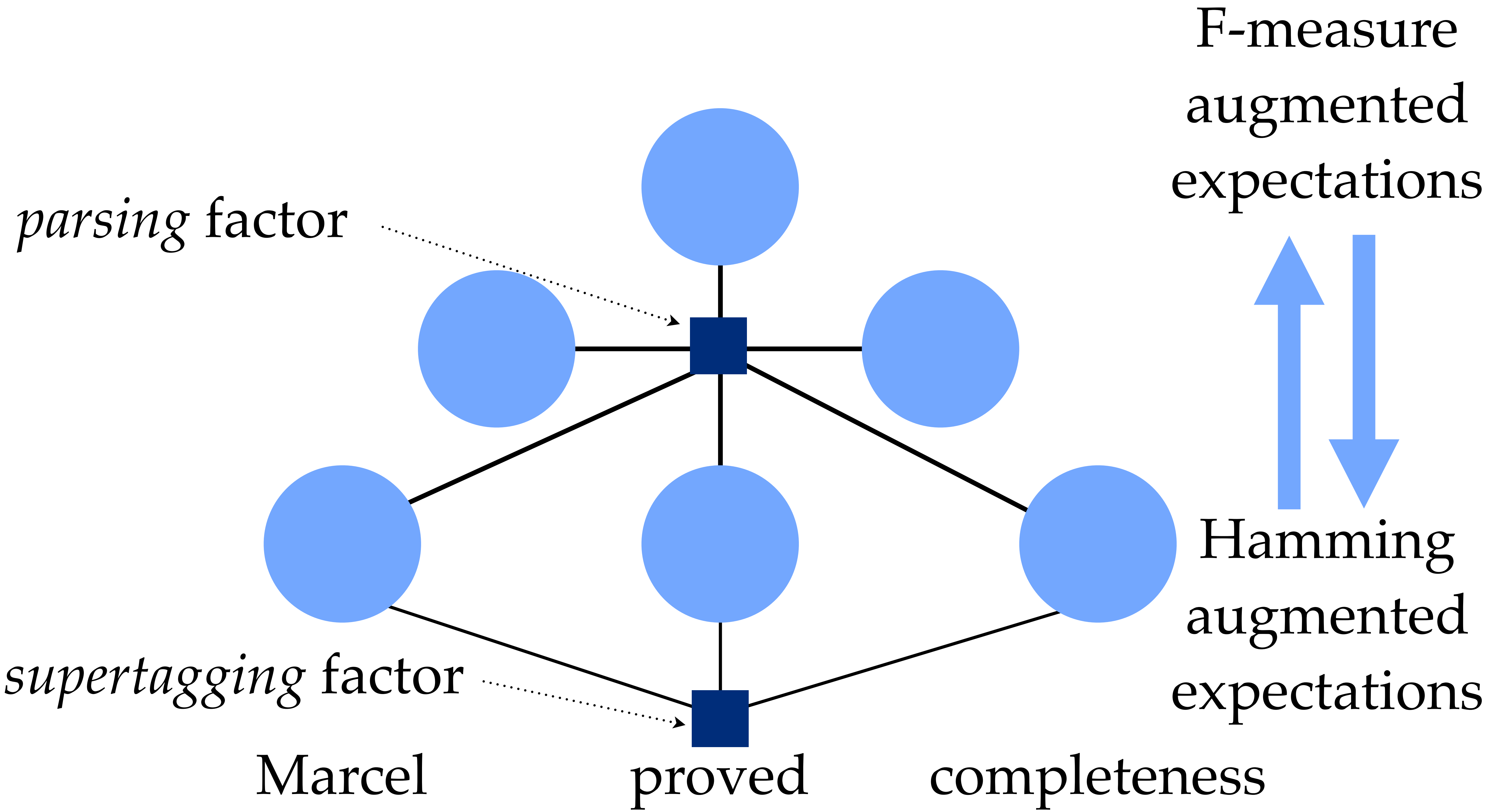
- Results on test:
- CLL
  - tight beam: 87.73
  - loose beam: 87.65
- DecF1
  - tight beam: 88.09
  - loose beam: **88.58**

Experiments: Baseline + SMM

- Results on test:
- CLL
  - tight beam: 87.73
  - loose beam: 87.65
- DecF1
  - tight beam: 88.09
  - loose beam: **88.58**

Best performance
in larger search space

Integrated Model + SMM



Experiments

- Results with integrated model using BP:

CLL	87.65

Experiments

- Results with integrated model using BP:

CLL	87.65
BP	88.86

Experiments

- Results with integrated model using BP:

CLL	87.65
BP	88.86
+ $DecF_1$	89.15

Experiments

- Results with integrated model using BP:

CLL	87.65	
BP	88.86	
+ DecF_1	89.15	parser loss
+ SA	**89.25**	Hamming loss

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74
Petrov I-5	86.01

Fowler & Penn (2010)

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74	
Petrov I-5	86.01	Fowler & Penn (2010)
BP	86.84	

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74	
Petrov 1-5	86.01	Fowler & Penn (2010)
BP	86.84	
+ DecF_1	87.08	

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74	
Petrov I-5	86.01	Fowler & Penn (2010)
BP	86.84	
+ $DecF_1$	87.08	parser loss
+ SA	**87.20**	Hamming loss

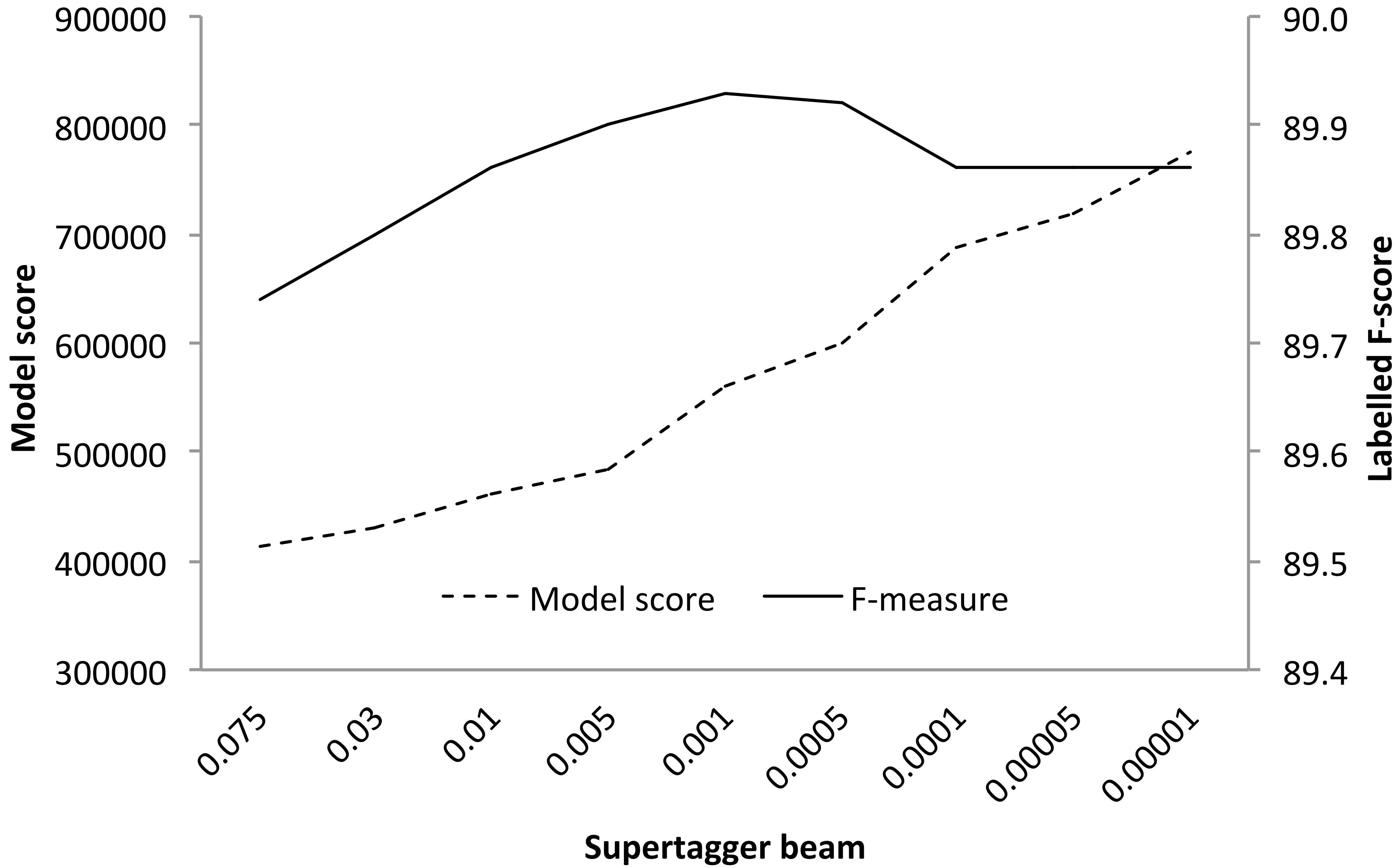
Experiments

- What is the performance when speed is important?
- Results with tight beam (AST):

	F_1	sent/sec	
CLL	87.73	65	
BP	88.20		
+ DecF_1	88.28	60	parser loss
+ SA	**88.46**		Hamming loss

Oracle Results Again

Belief Propagation



Conclusion

- Pruning with supertagging comes at a cost.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.
- Supertagger and parser interaction can be exploited in a combined model.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.
- Supertagger and parser interaction can be exploited in a combined model.
- Task specific optimization improves performance.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.
- Supertagger and parser interaction can be exploited in a combined model.
- Task specific optimization improves performance.
- Approximate loss functions very competitive to exact counterparts -- they are simple and efficient.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.
- Supertagger and parser interaction can be exploited in a combined model.
- Task specific optimization improves performance.
- Approximate loss functions very competitive to exact counterparts -- they are simple and efficient.
- Methods are generally applicable.

Conclusion

- Pruning with supertagging comes at a cost.
- Better models can exploit larger search spaces.
- Supertagger and parser interaction can be exploited in a combined model.
- Task specific optimization improves performance.
- Approximate loss functions very competitive to exact counterparts -- they are simple and efficient.
- Methods are generally applicable.
- **Best reported results for CCG parsing (89.3)**

Future Directions

- Integration of POS sequence model.
- Combined models for grammar induction.
- Application to other grammar formalisms & problems.

Thanks!

Phil Blunsom
Prachya Boonkwan
Christos Christodoulopoulos
Stephen Clark
Michael Collins
Chris Dyer
Timothy Fowler
Mark Granroth-Wilding
Philipp Koehn
Terry Koo
Tom Kwiatkowski
André F. T. Martins
Matt Post
Gholamreza Haffari
David A. Smith
David Sontag
Mark Steedman
Charles Sutton