Accurate CCG Parsing with Approximate Language Intersection and Task-specific Optimization

Michael Auli
joint work with Adam Lopez (Johns Hopkins University)

Parsing

Parsing

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

Marcel proved completeness

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

language-specific
information in *lexicon*

Marcel proved completeness

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

language-specific
information in *lexicon*

Marcel	proved	completeness
NP	$(S\backslash NP)/NP$	NP

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

language-specific
information in *lexicon*

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

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

language-specific
information in *lexicon*

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

Combinatory Categorial Grammar (CCG; Steedman 2000)

Parsing

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

Parsing

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

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

Hard parsing task

Overview

- Part I: Search in Lexicalized Grammar Parsing
  Pruning and Optimality
- Part II: More Accurate Search with Combined Models
  with Loopy Belief Propagation and Dual Decomposition (Auli & Lopez 2011)
- Part III: Task-specific Optimization
  with Softmax-Margin using Exact and Approximate Loss Functions

Overview

- Part I: Search in Lexicalized Grammar Parsing
  Pruning and Optimality
- Part II: More Accurate Search with Combined Models
  with Loopy Belief Propagation and Dual Decomposition (Auli & Lopez 2011)
- Part III: Task-specific Optimization
  with Softmax-Margin using Exact and Approximate Loss Functions

Supertagging

Supertagging

time flies like an arrow

Supertagging

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

Supertagging

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

$$\underline{\qquad\qquad\qquad}>$$
$$\mathrm{NP}$$

$$\underline{\qquad\qquad\qquad}>$$
$$(\mathrm{S}\backslash\mathrm{NP})\backslash(\mathrm{S}\backslash\mathrm{NP})$$

$$\underline{\qquad\qquad\qquad}<$$
$$\mathrm{S}\backslash\mathrm{NP}$$

$$\underline{\qquad\qquad\qquad}<$$
$$\mathrm{S}$$

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

$$\xrightarrow{\quad\quad\quad} > \quad \mathrm{NP}$$

$$\xrightarrow{\quad\quad\quad} > \quad \mathrm{S\backslash NP}$$

Supertagging

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

$$\overline{}\,>$$

NP

$$\overline{}\,>$$

$S\backslash NP$

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

Adaptive Supertagging

- 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

- 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...

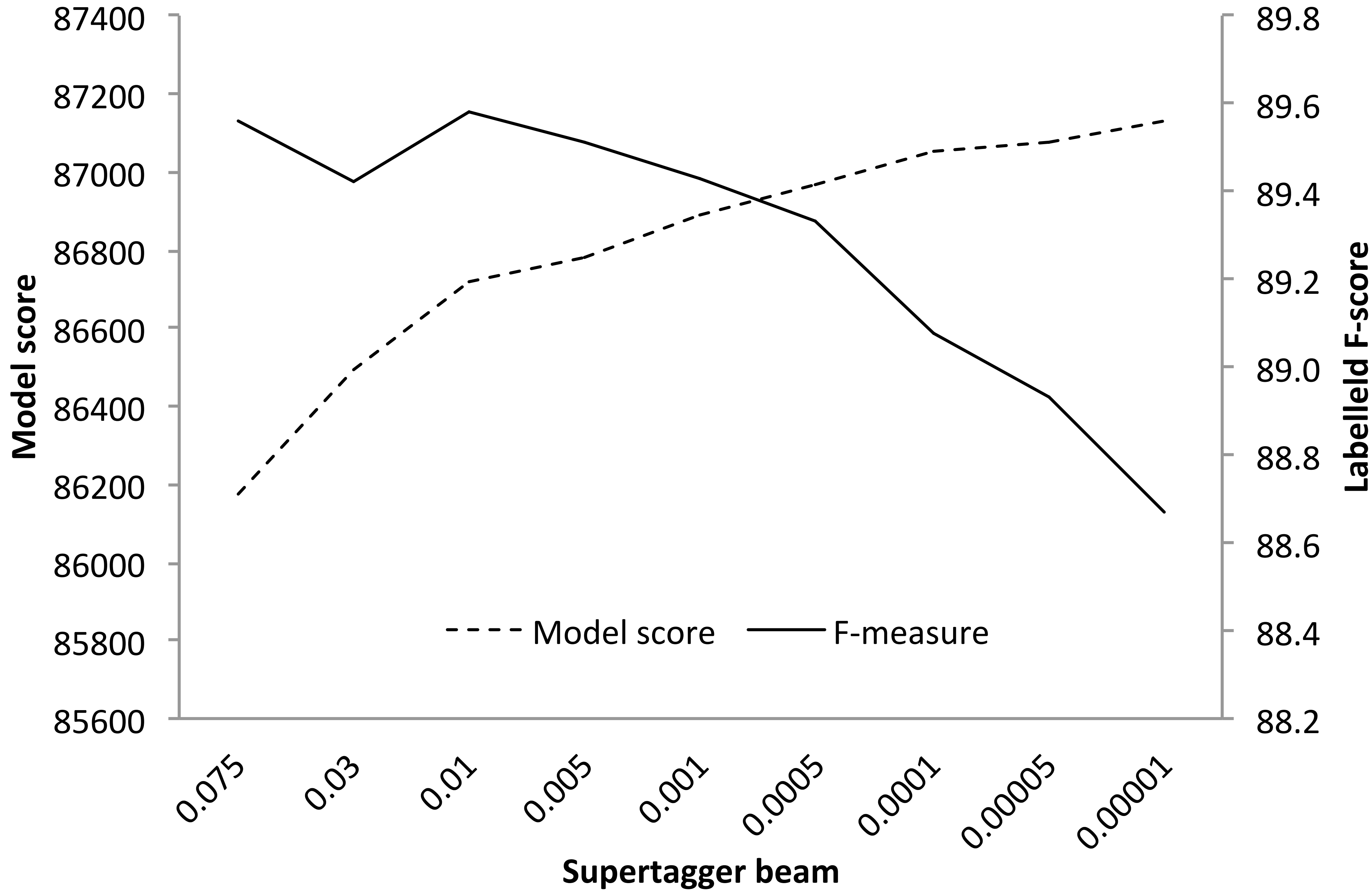
Answer...

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

Answer...

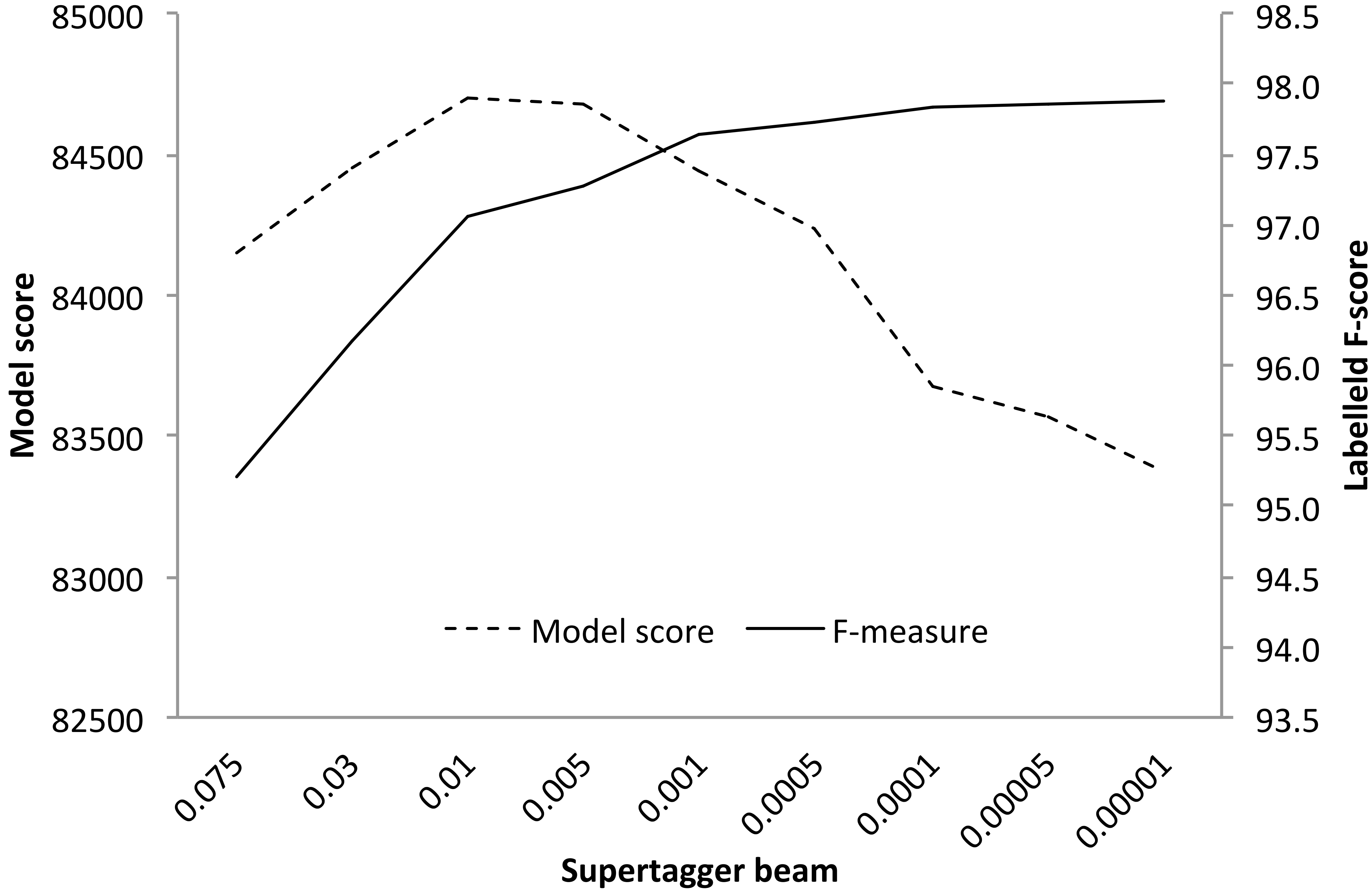
- Oracle parsing (Huang 2008):
  - With tight beam: 94.35
  - With loose beam: **97.65**
- Standard parsing task (Clark & Curran 2007):
  - With tight (*adaptive*) beam: 87.38 (labeled F-measure)
  - With loose (*reverse*) beam: 87.36

Oracle 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

- Part I: Search in Lexicalized Grammar Parsing
  Pruning and Optimality
- Part II: More Accurate Search with Combined Models
  with Loopy Belief Propagation and Dual Decomposition (Auli & Lopez 2011)
- Part III: Task-specific Optimization
  with Softmax-Margin using Exact and Approximate Loss Functions

Integrated Model

- Supertagger and parser are both undirected 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 and parser are both undirected 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

- Supertagger and parser are both undirected 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.

original parsing problem: A → B C $O(Gn^3)$

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Integrated Model

- Supertagger and parser are both undirected 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.

original parsing problem: $A \rightarrow B\ C$ $O(Gn^3)$

new parsing problem: ${}_{q}A_{r} \rightarrow {}_{q}B_{s}\ {}_{s}C_{r}$ $O(G^3n^3)$

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Approximate Algorithms

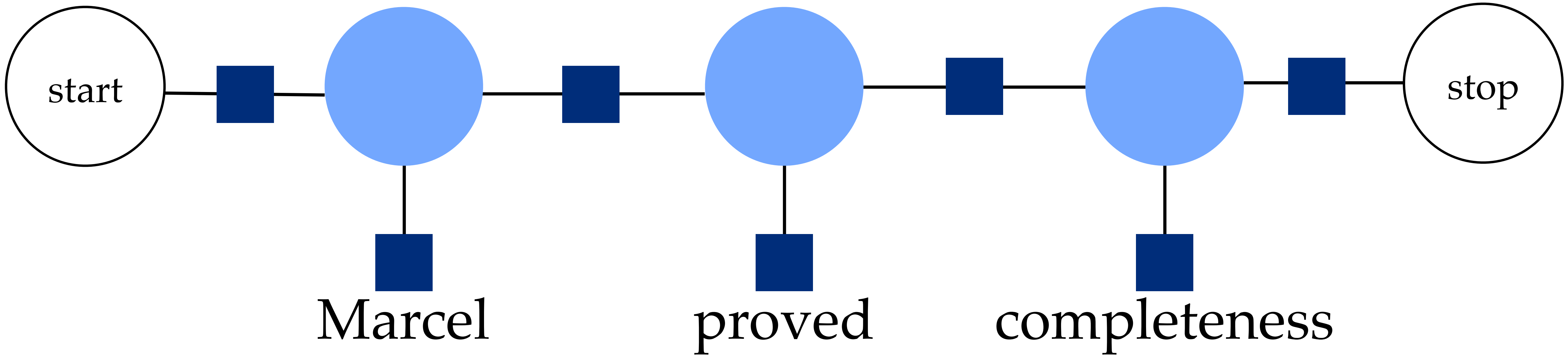
- Loopy belief propagation: approximate calculation of marginals. (Pearl 1988; Smith & Eisner 2008)
- Dual decomposition: exact (sometimes) calculation of maximum. (Dantzig & Wolfe 1960; Komodakis et al. 2007; Koo et al. 2010)

Belief Propagation

Belief Propagation

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

Belief Propagation



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

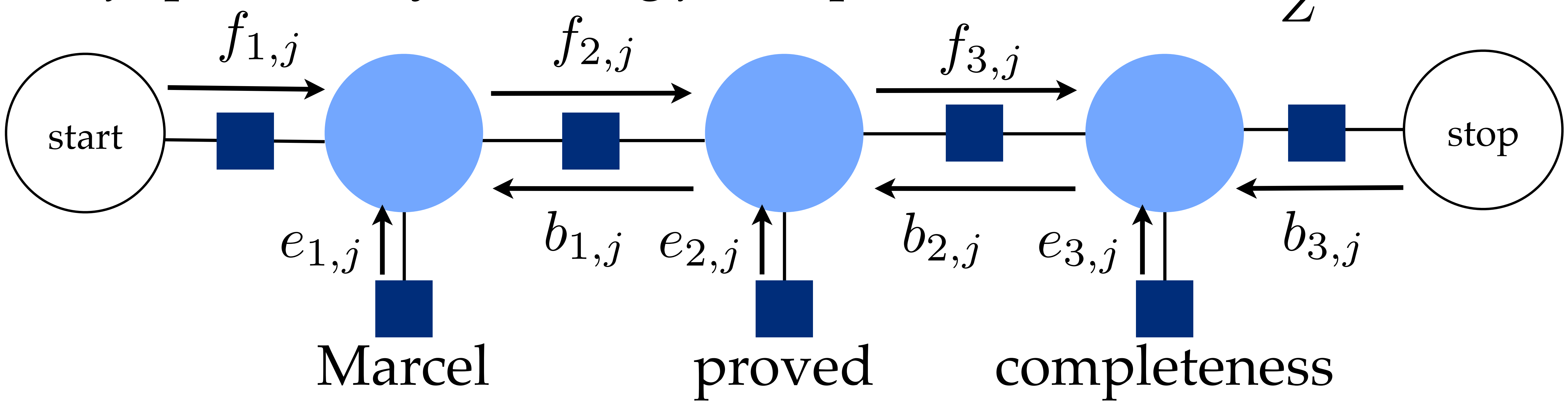
Belief Propagation

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

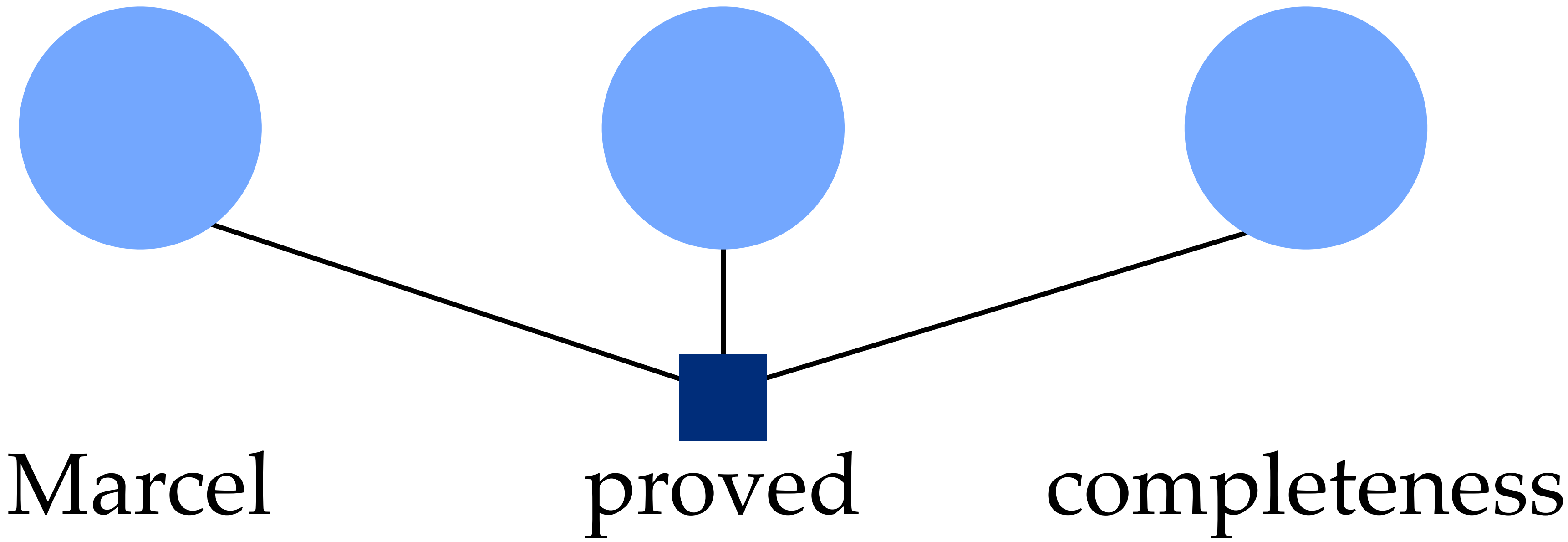
belief (probability) that tag *j* is at position *i*: $p_{i,j} = \frac{1}{Z} f_{i,j} e_{i,j} b_{i,j}$



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

Belief Propagation

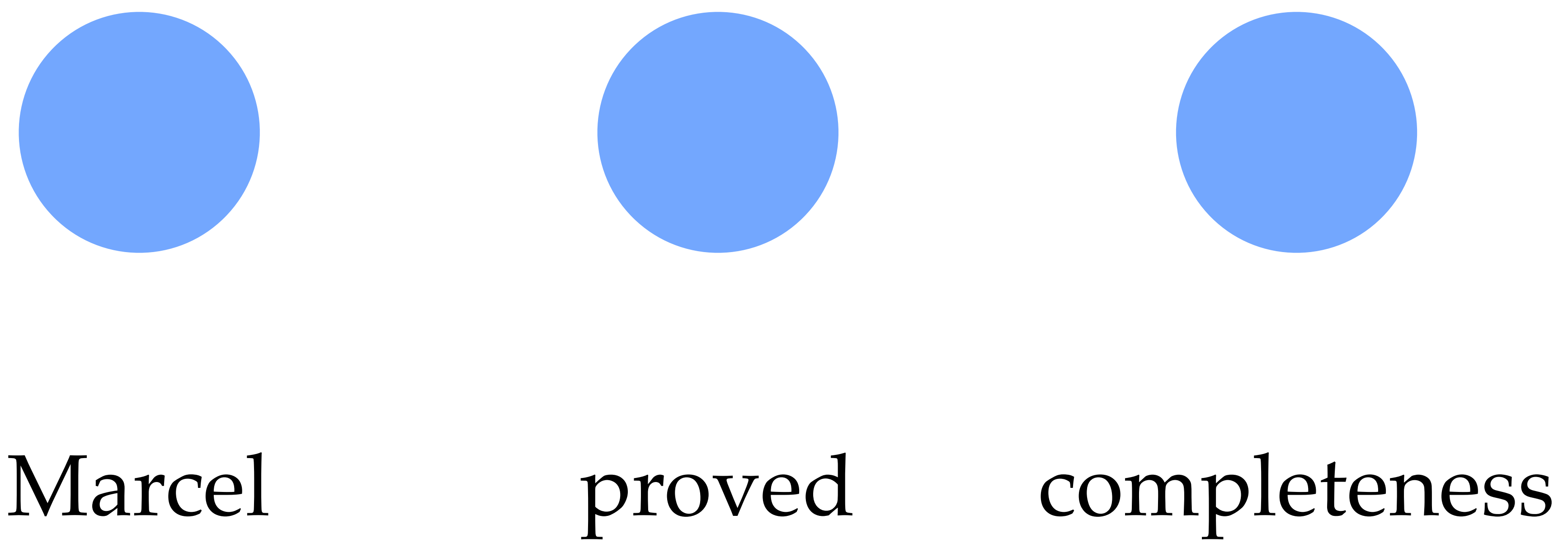
Notational convenience: one factor describes whole distribution over supertag sequence...



Belief Propagation

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

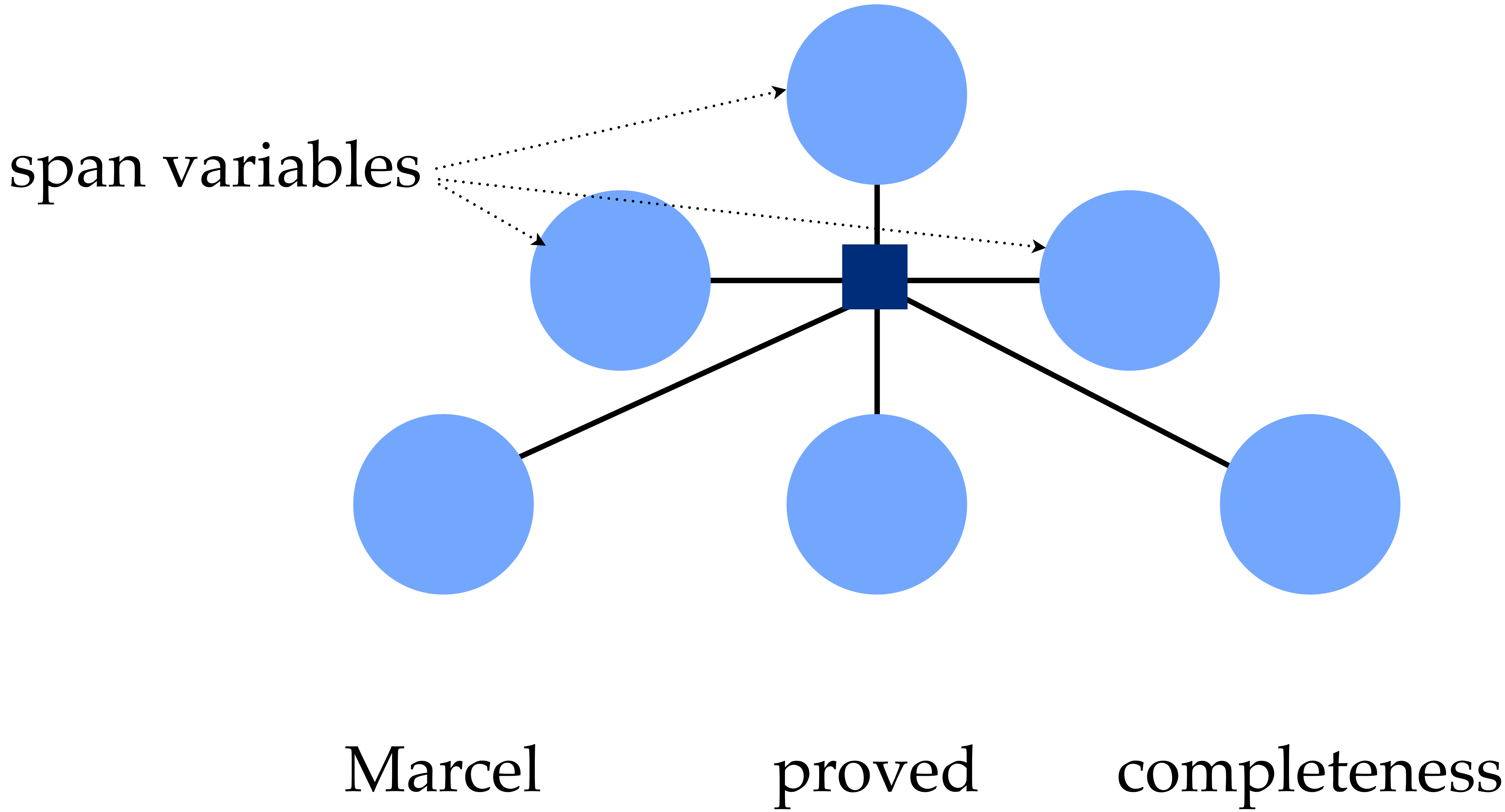
(Case-factor diagrams: McAllester et al. 2008)



Belief Propagation

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

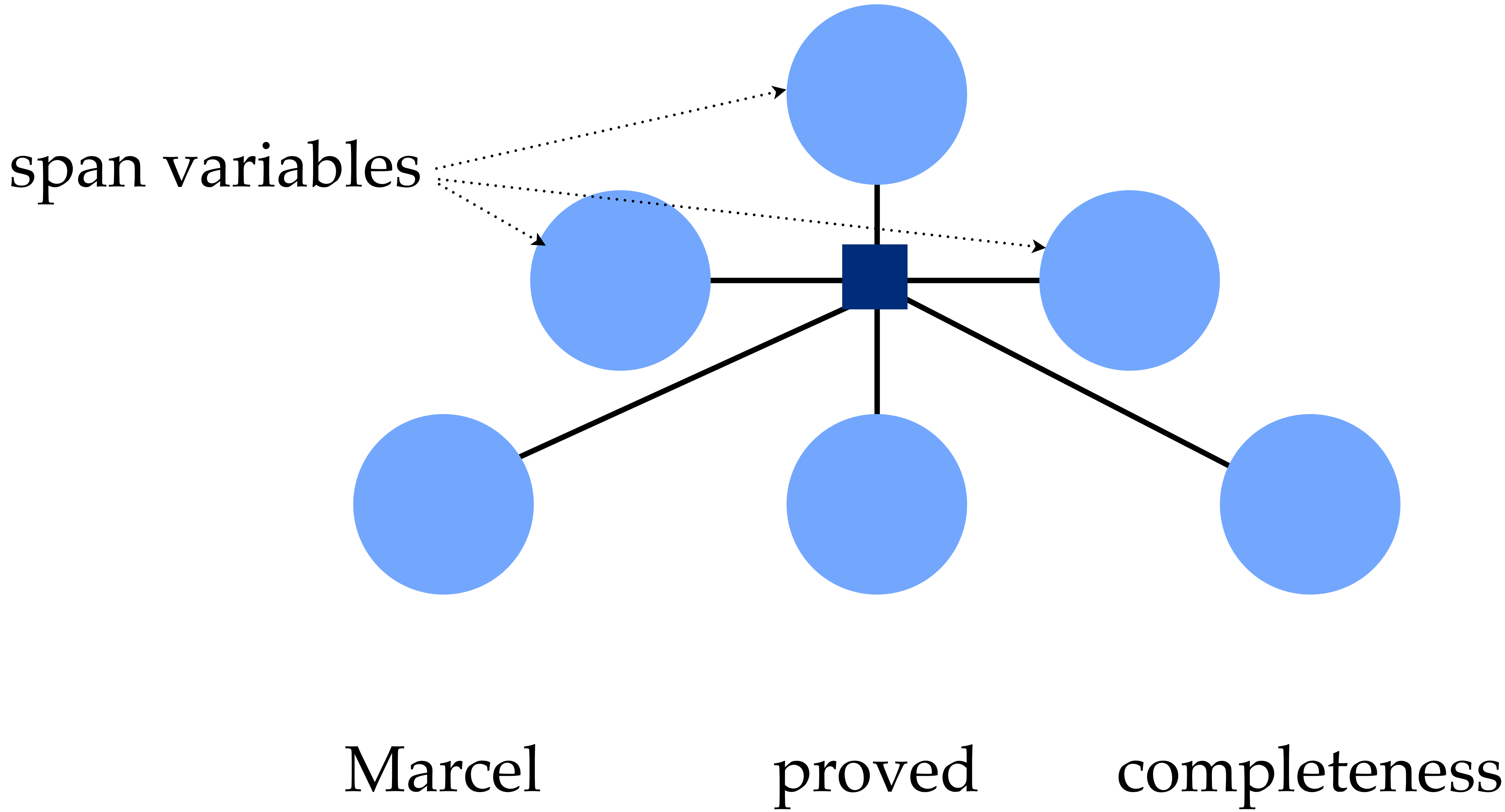
(Case-factor diagrams: McAllester et al. 2008)



Belief Propagation

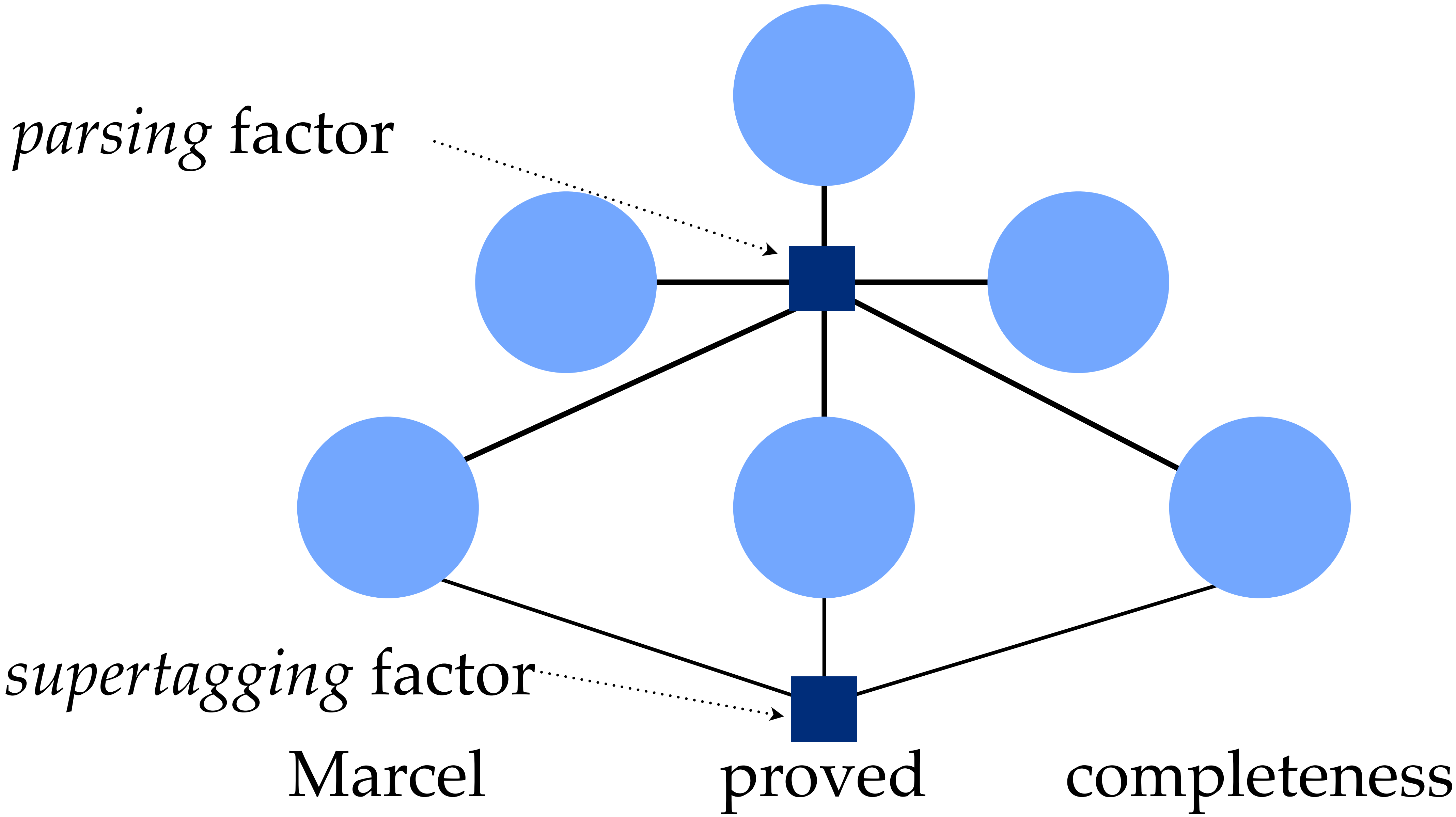
We can also do the same for the distribution over parse trees

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



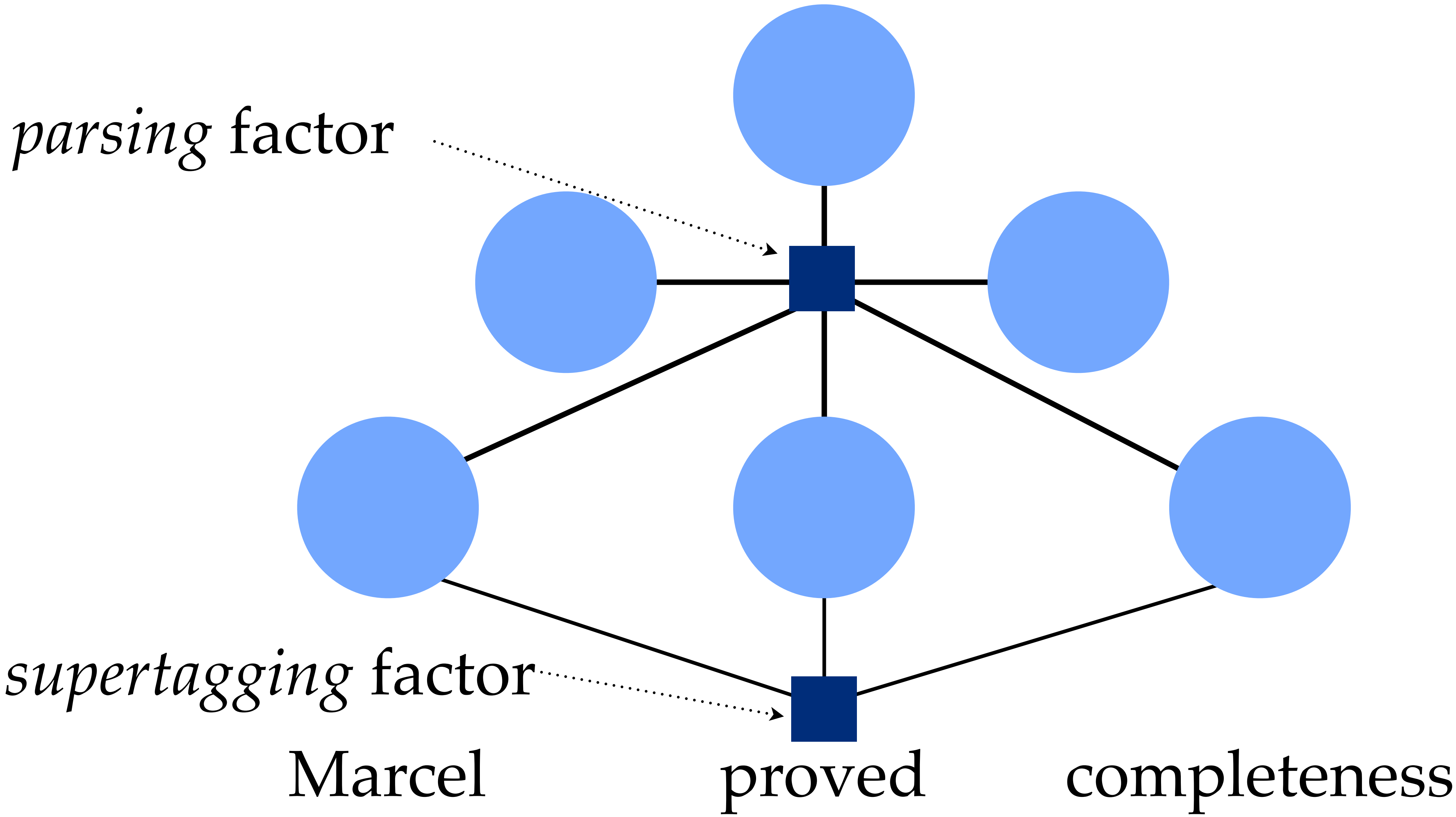
Inside-outside is belief propagation (Sato 2007)

Belief Propagation



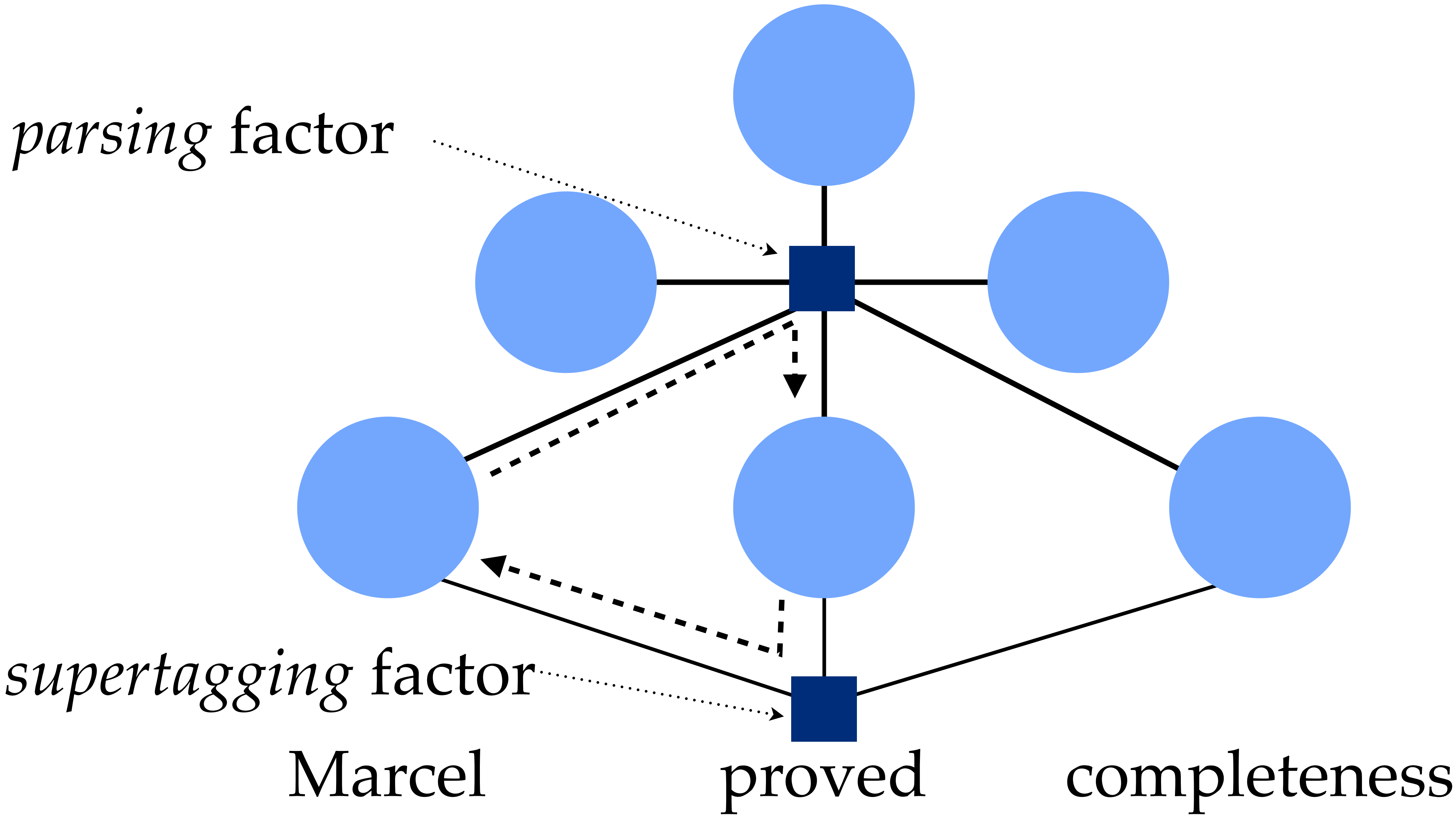
Belief Propagation

Graph is not a tree!



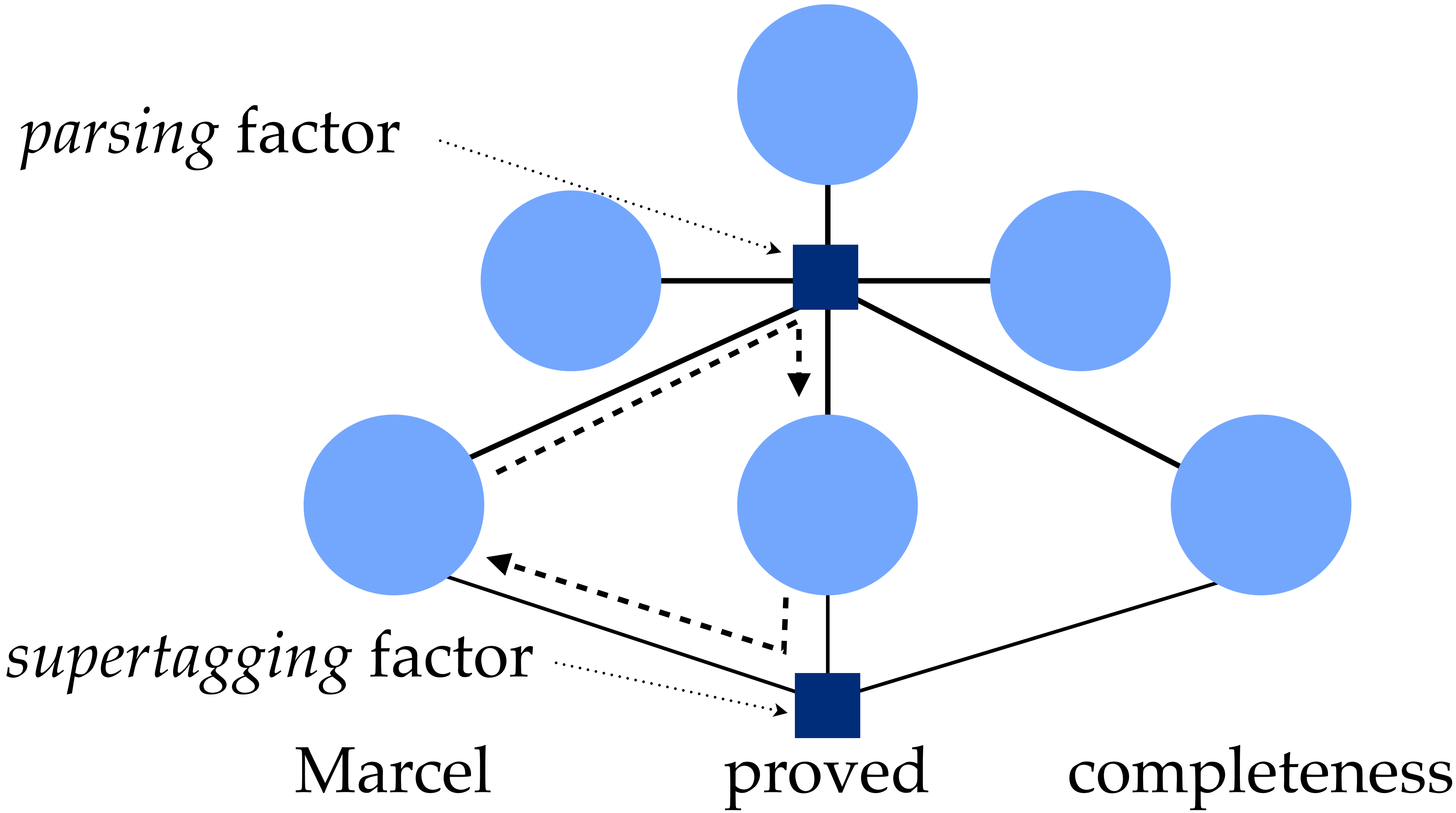
Belief Propagation

Graph is not a tree!



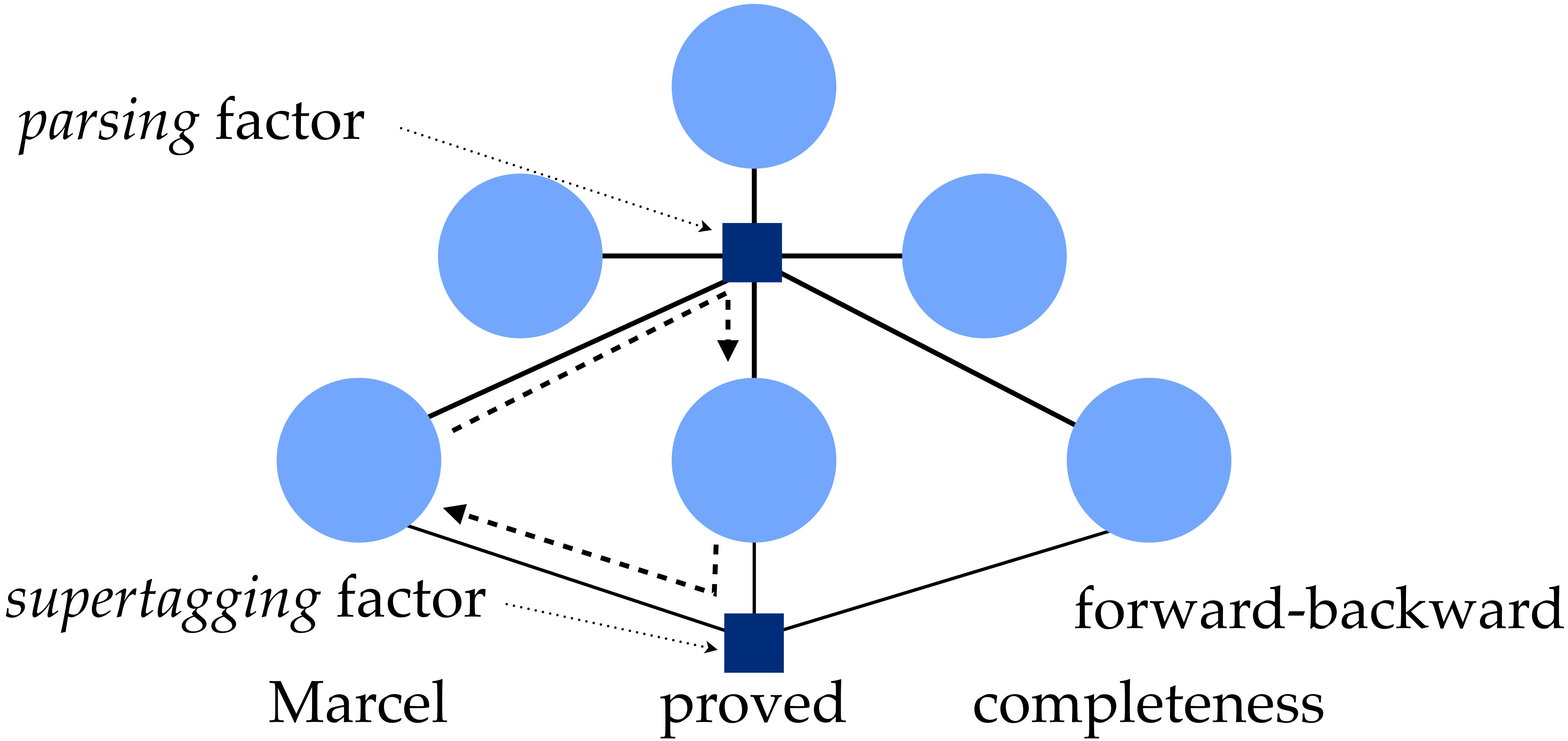
Loopy Belief Propagation

Graph is not a tree!



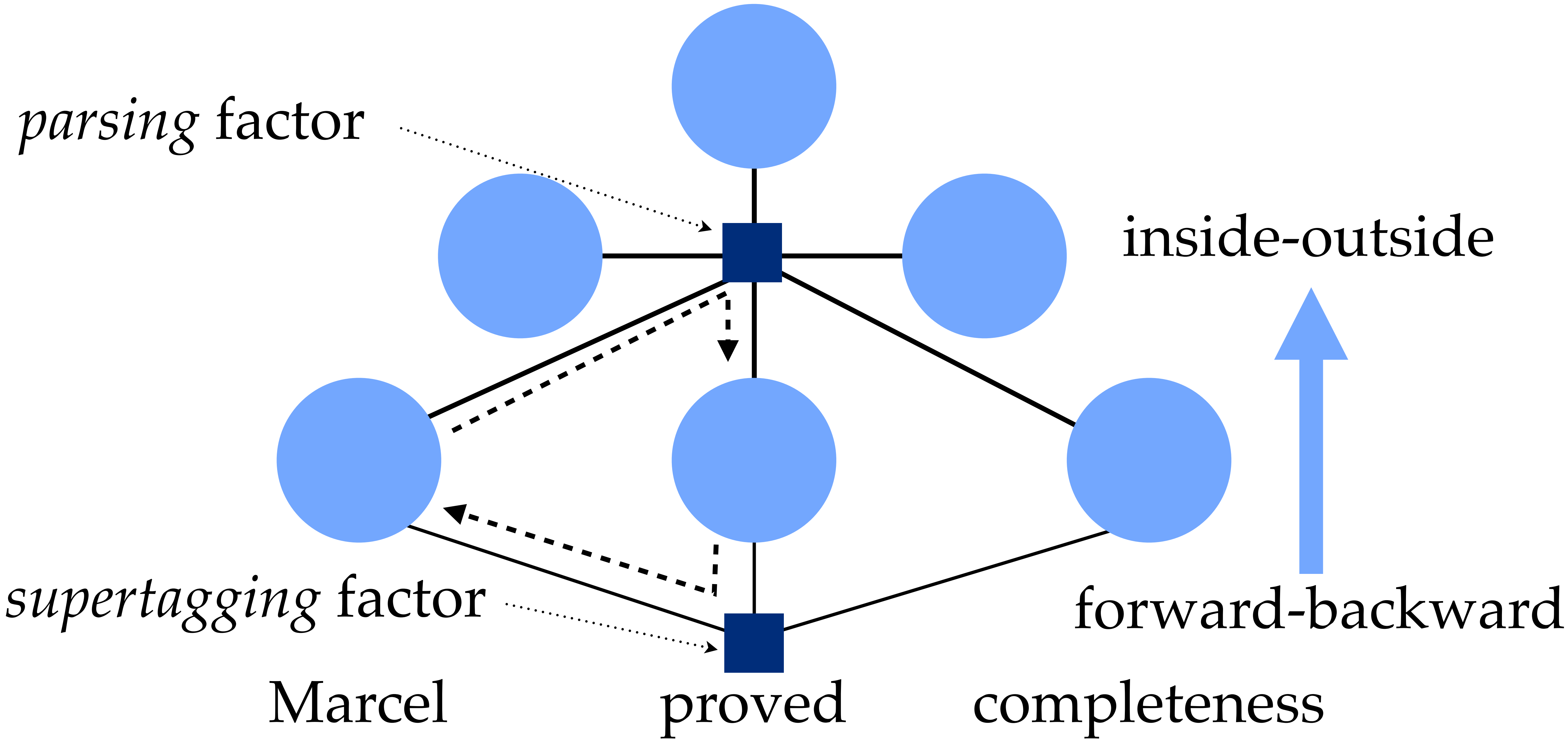
Loopy Belief Propagation

Graph is not a tree!



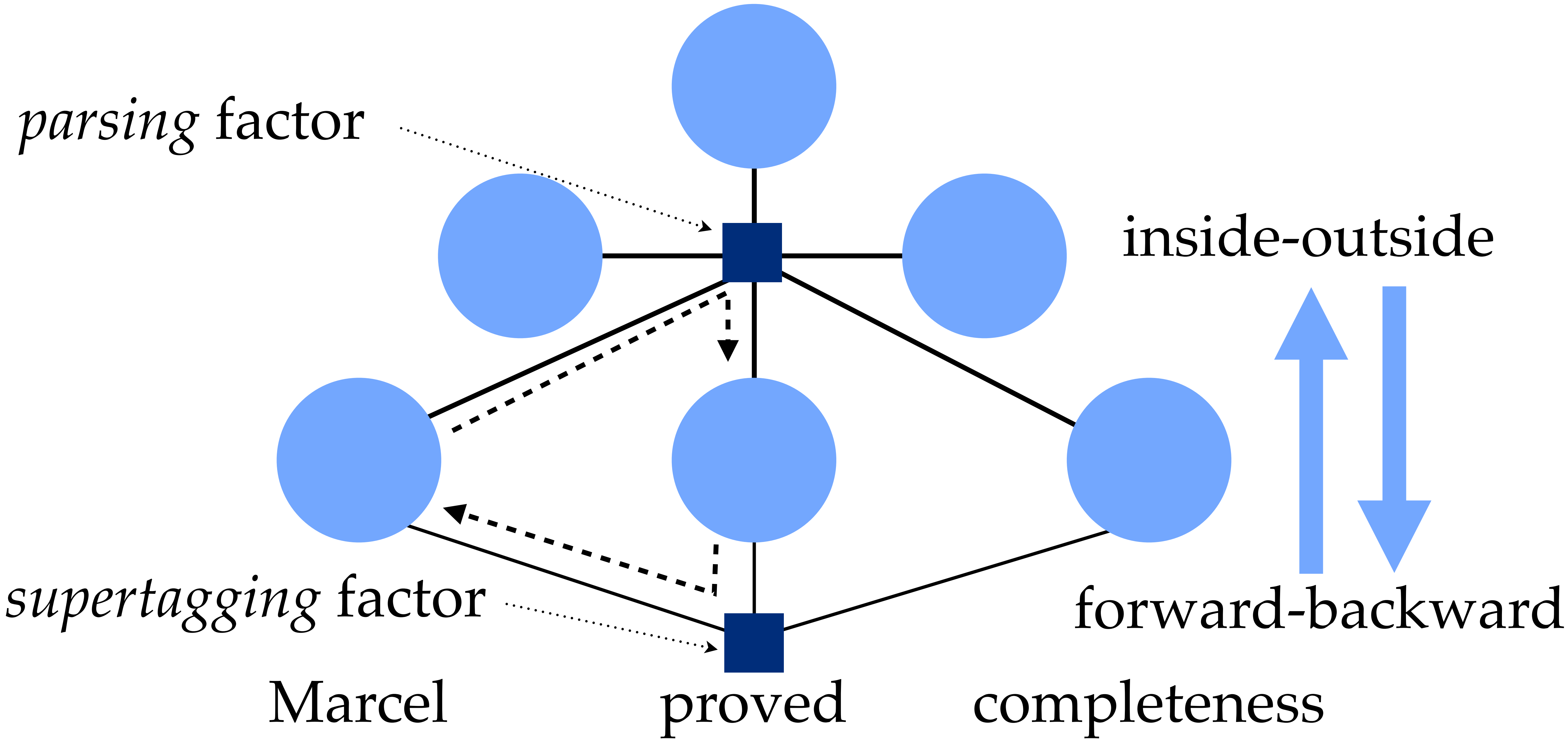
Loopy Belief Propagation

Graph is not a tree!



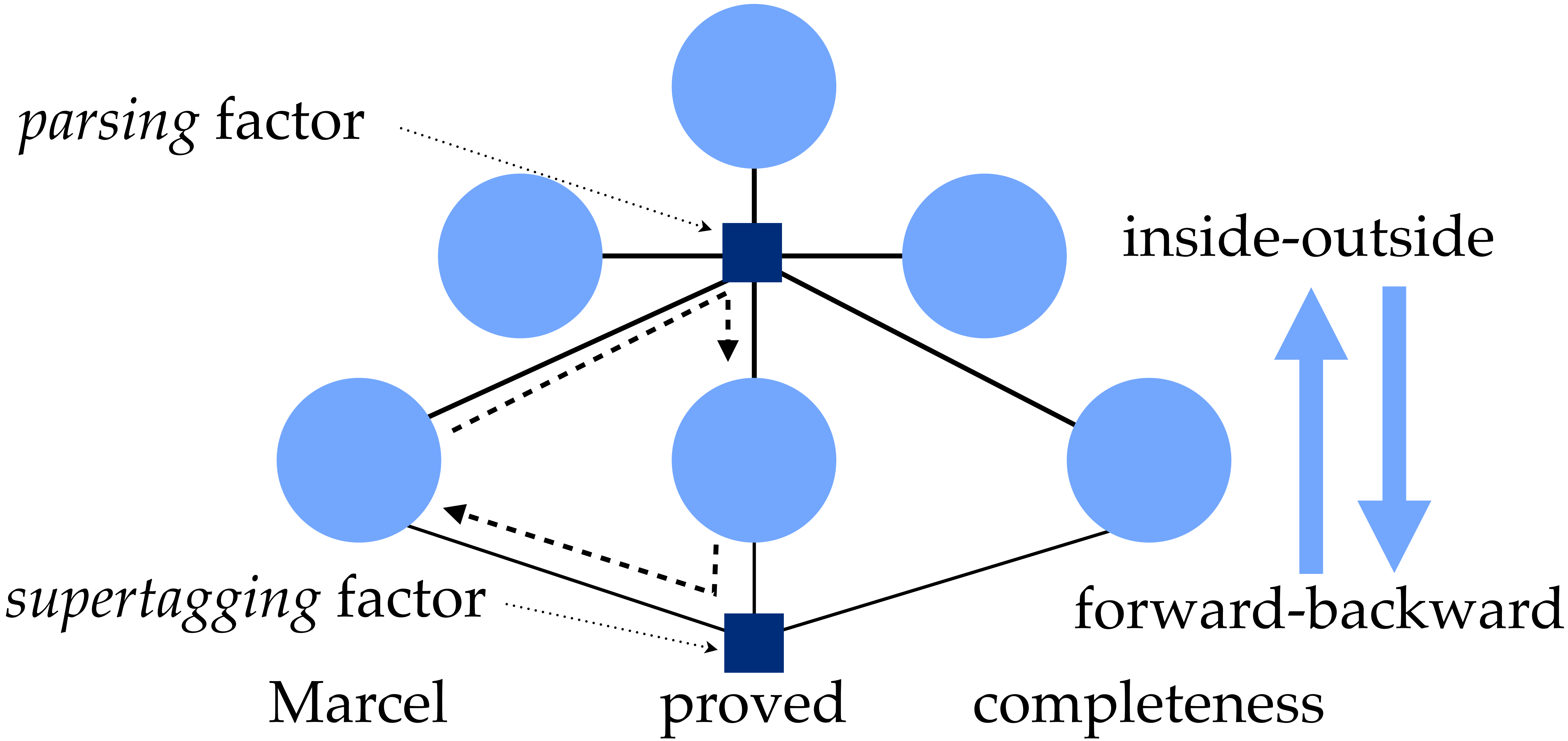
Loopy Belief Propagation

Graph is not a tree!



Loopy Belief Propagation

Graph is not a tree!

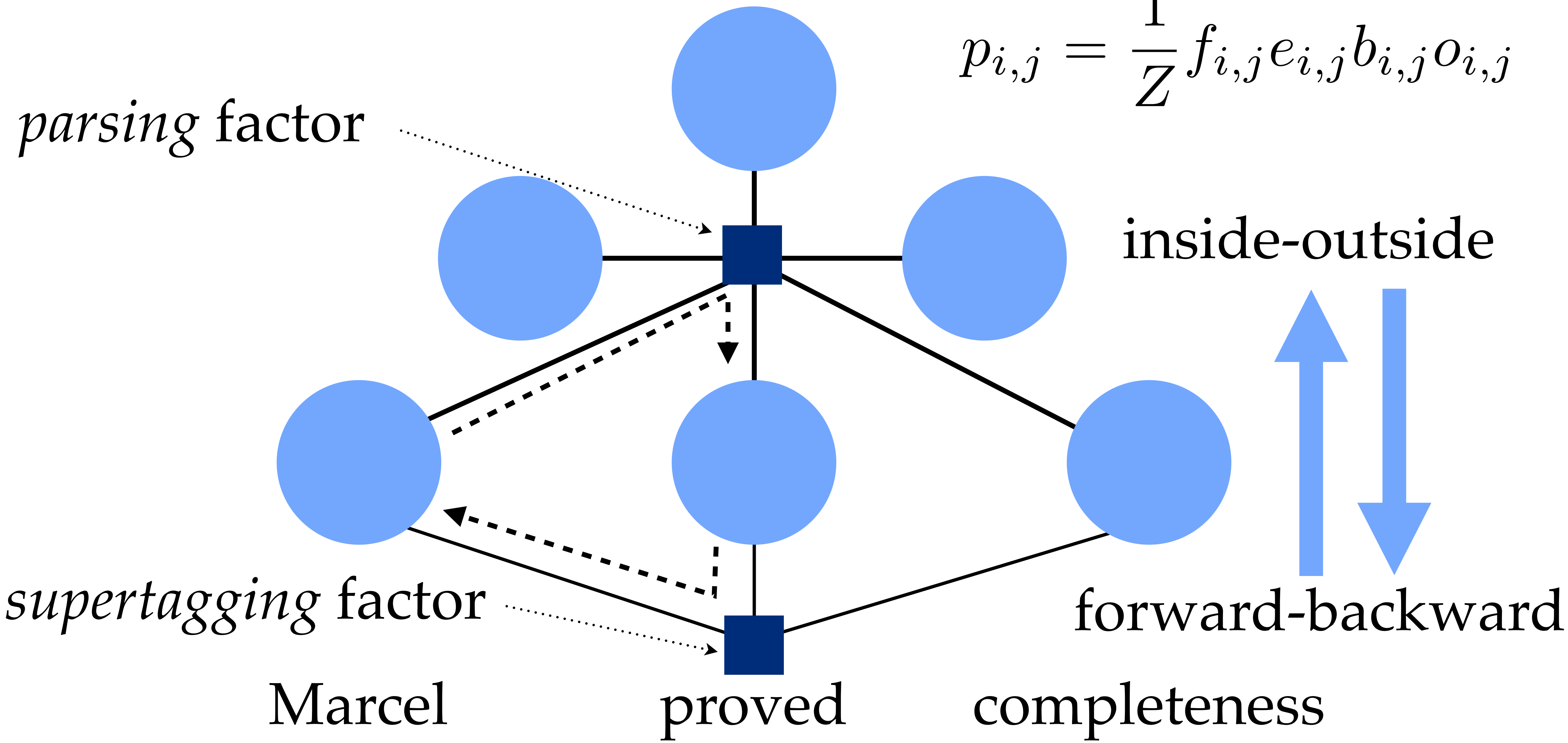


Converges to bounded approximate marginals (Yedidia et al. 2001)

Loopy Belief Propagation

Graph is not a tree!

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

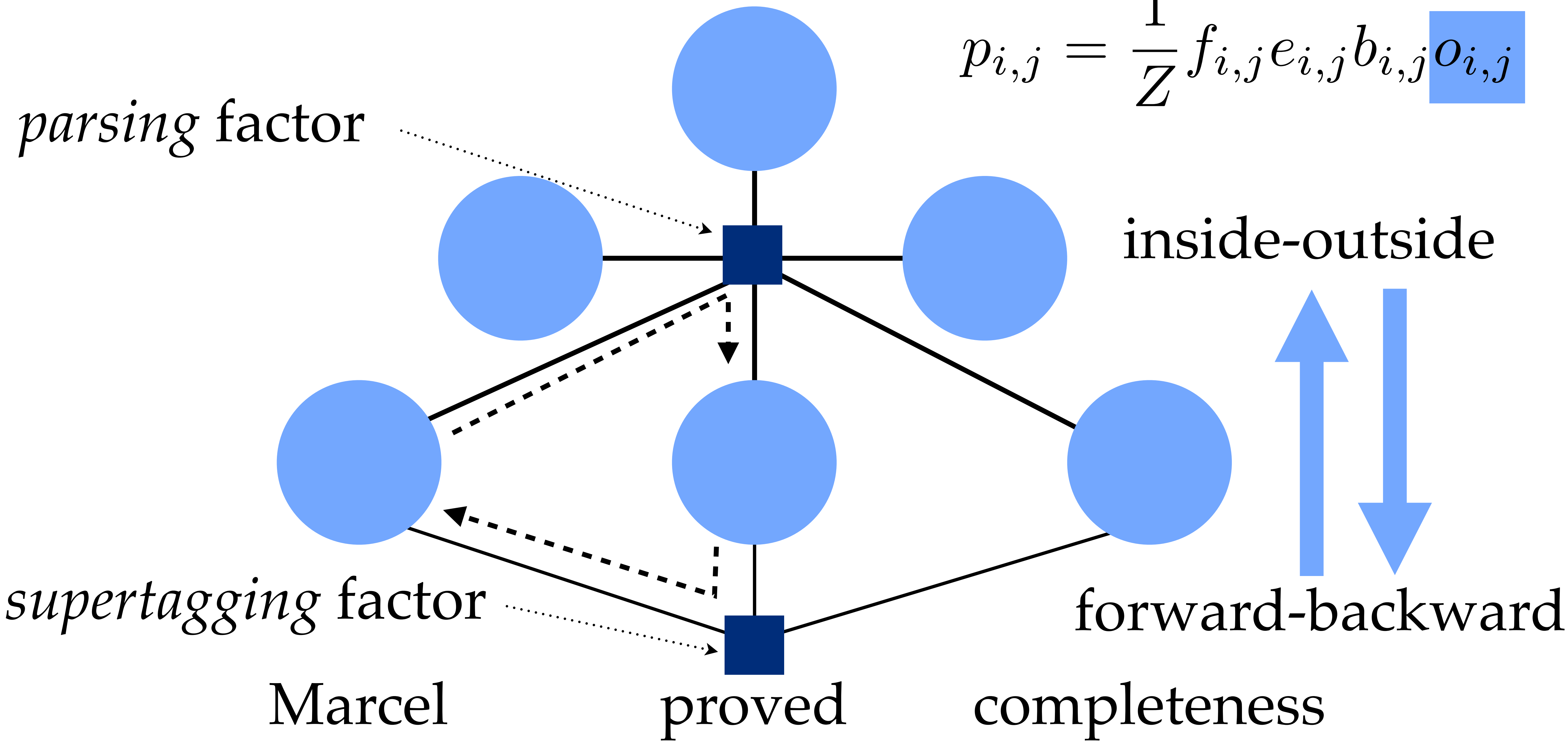


Converges to bounded approximate marginals (Yedidia et al. 2001)

Loopy Belief Propagation

Graph is not a tree!

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

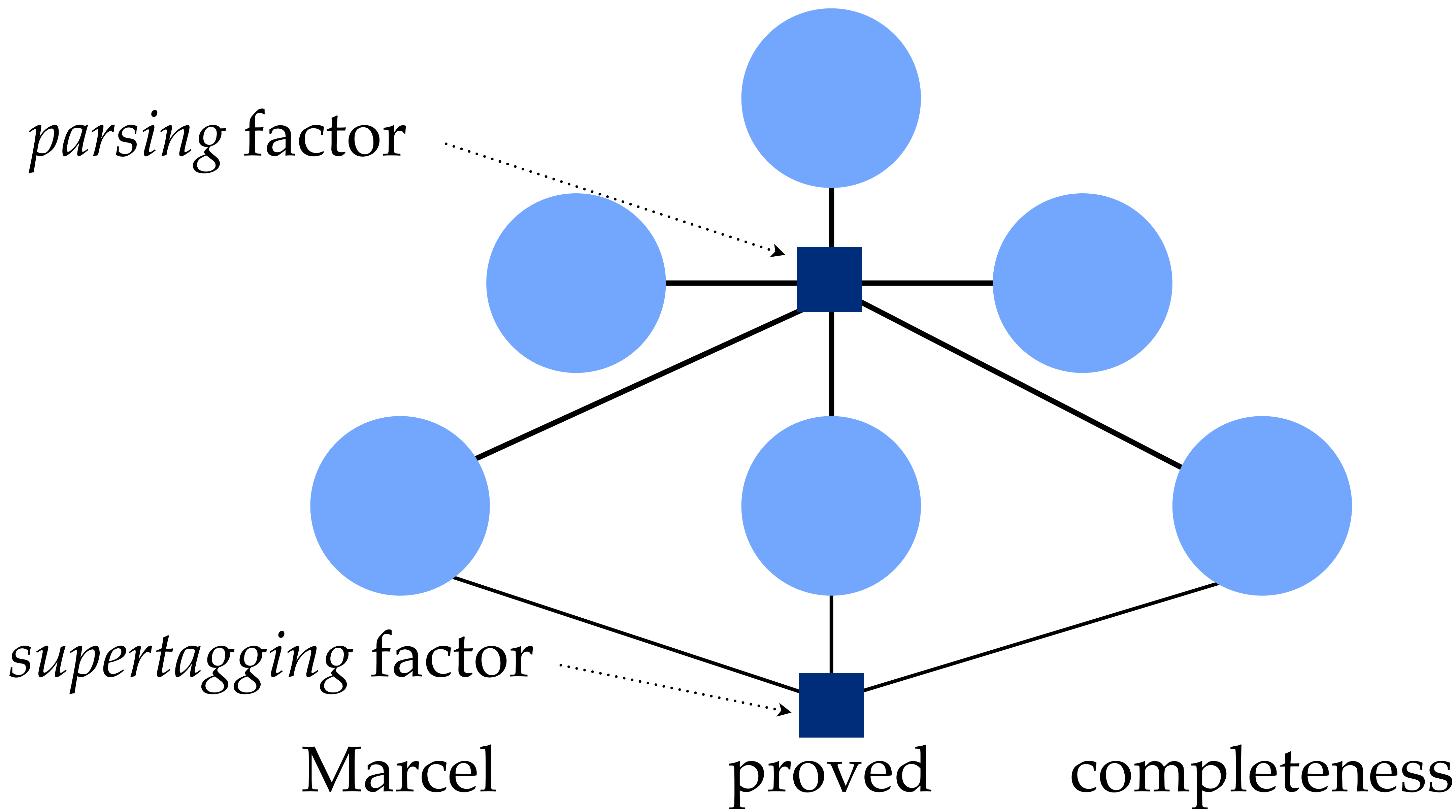


Converges to bounded approximate marginals (Yedidia et al. 2001)

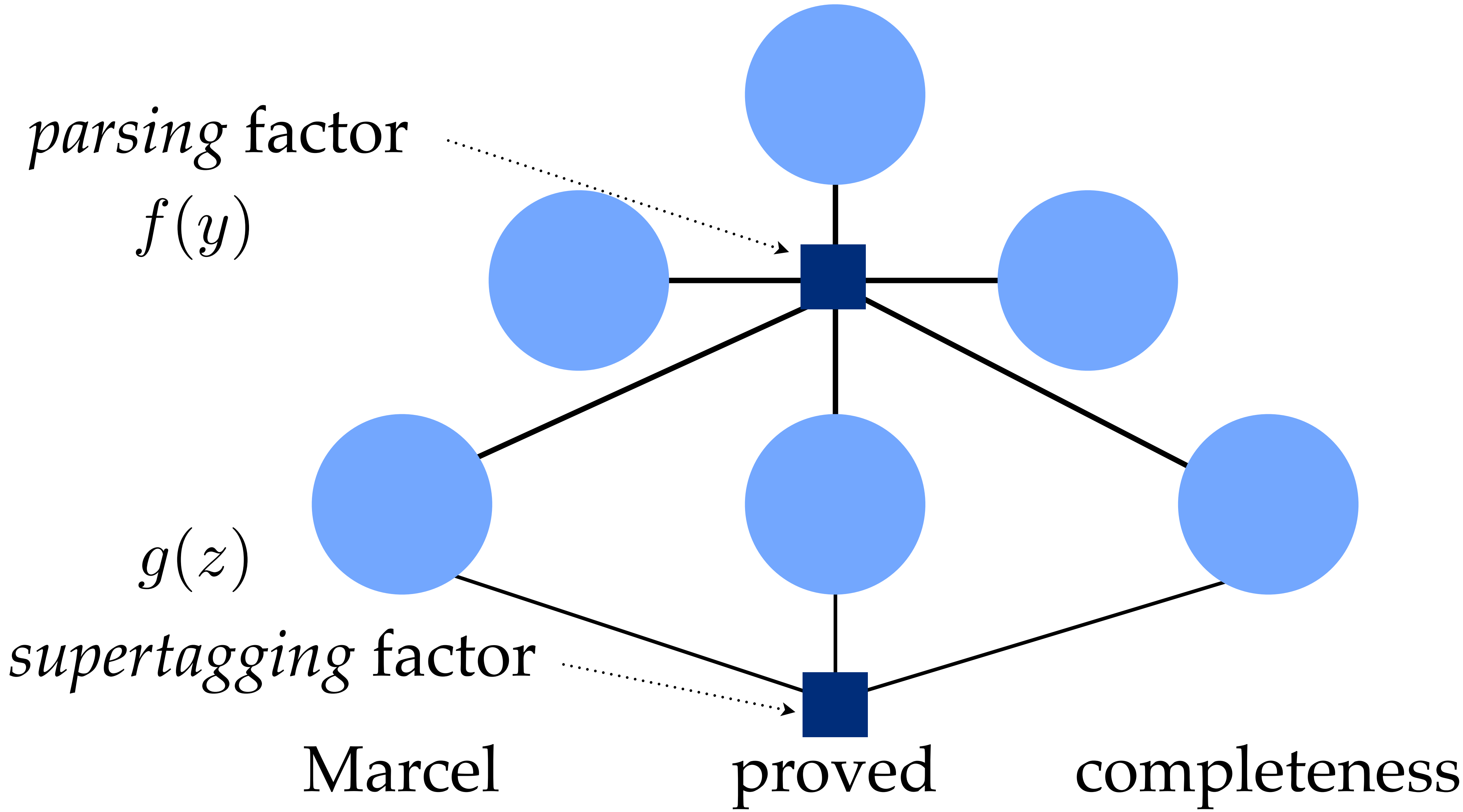
Loopy Belief Propagation

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

Dual Decomposition

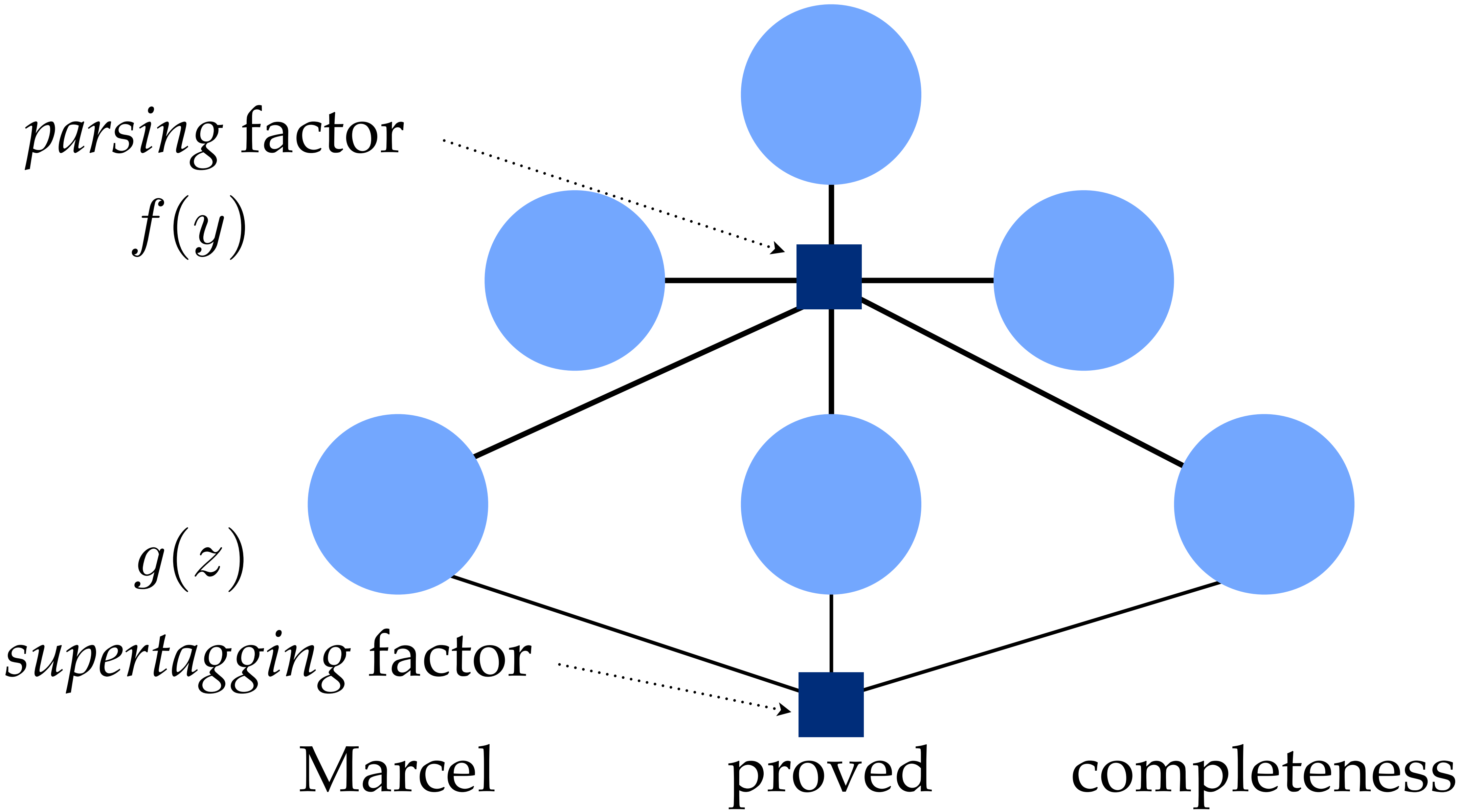


Dual Decomposition



Dual Decomposition

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



Dual Decomposition

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

Dual Decomposition

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

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

Dual Decomposition

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

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

original
problem

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

Dual Decomposition

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

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

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

Dual Decomposition

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

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

$u(i,t)$ (Lagrange multipliers) are messages!

Dual Decomposition

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

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

$u(i,t)$ (Lagrange multipliers) are messages!

$$u(i,t) = u(i,t) + \alpha \cdot [y(i,t) - z(i,t)]$$

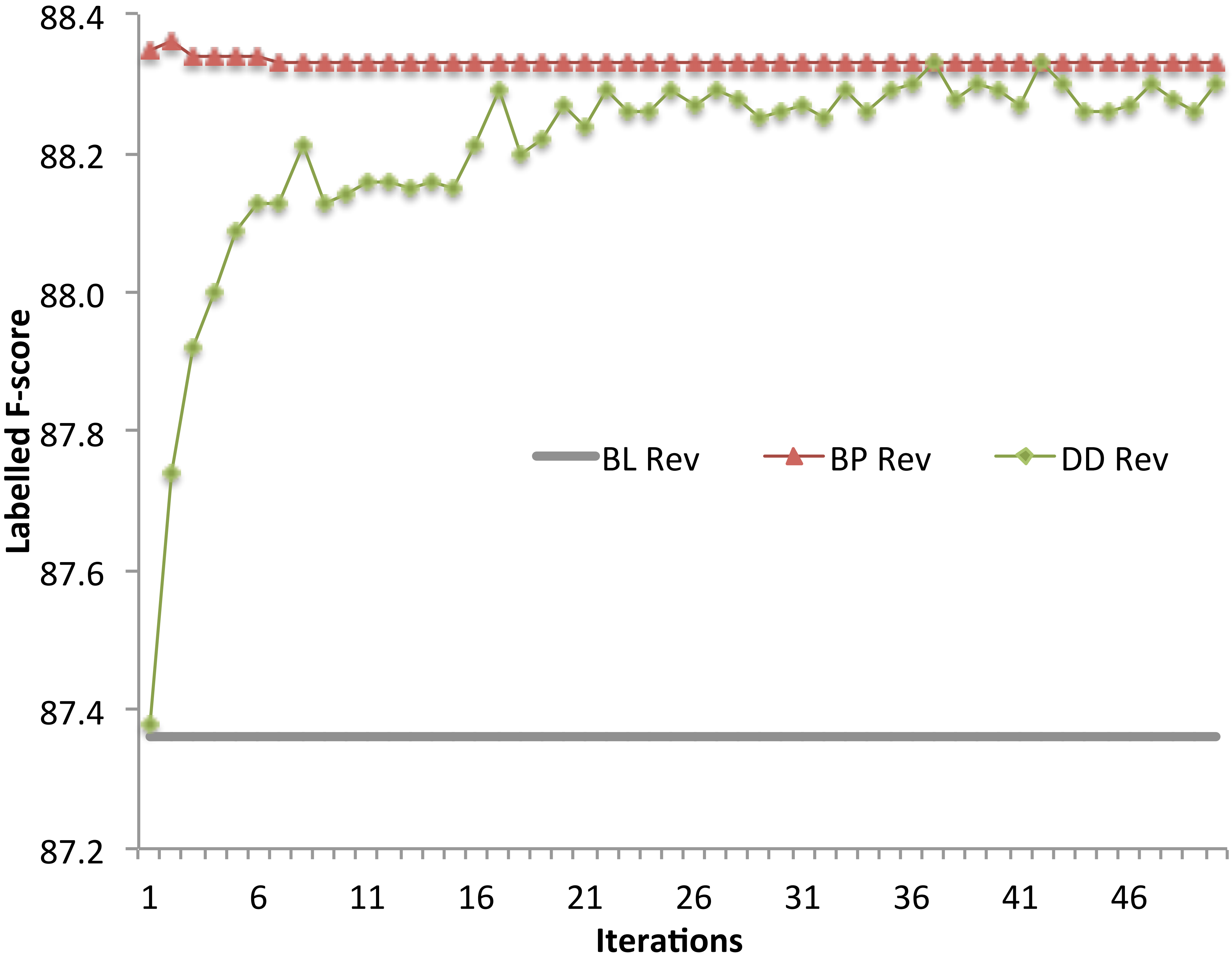
Dual Decomposition

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

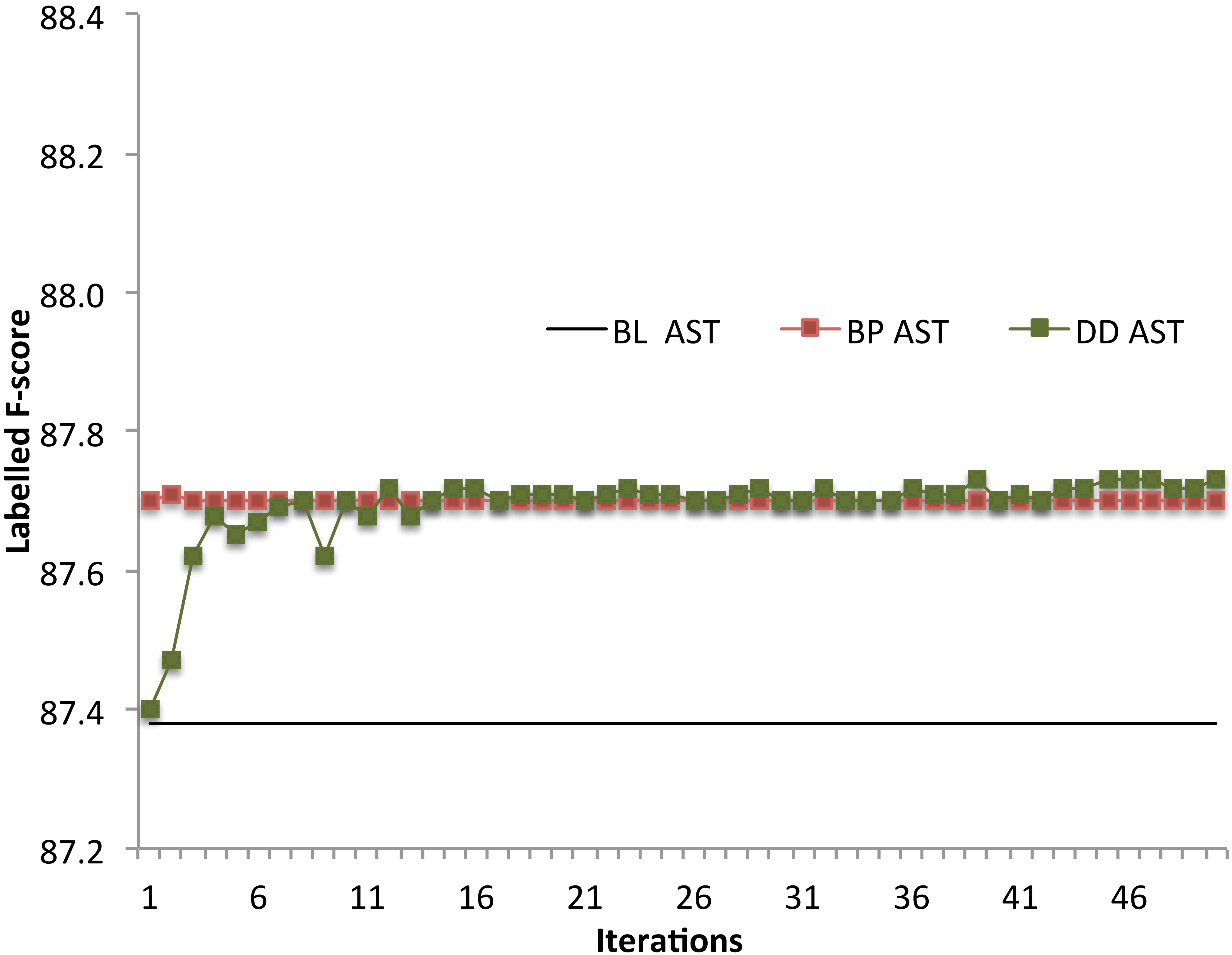
Experiments

- 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).
  - Features as in baseline: dependency-features, trigram features etc.
  - Approximate algorithms used to decode test set.

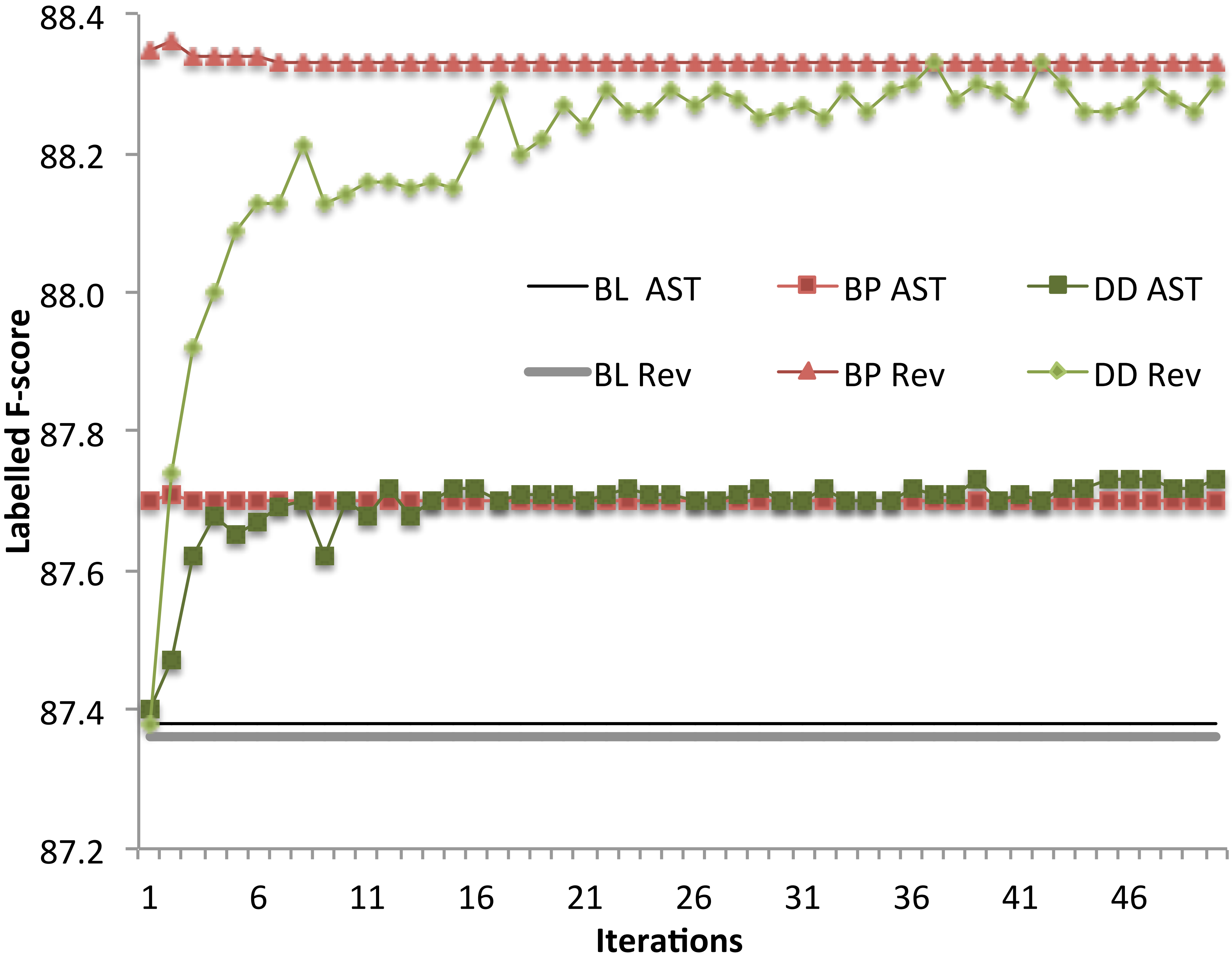
Experiments



Experiments



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: 88.19
  - loose beam: **88.80**

Experiments

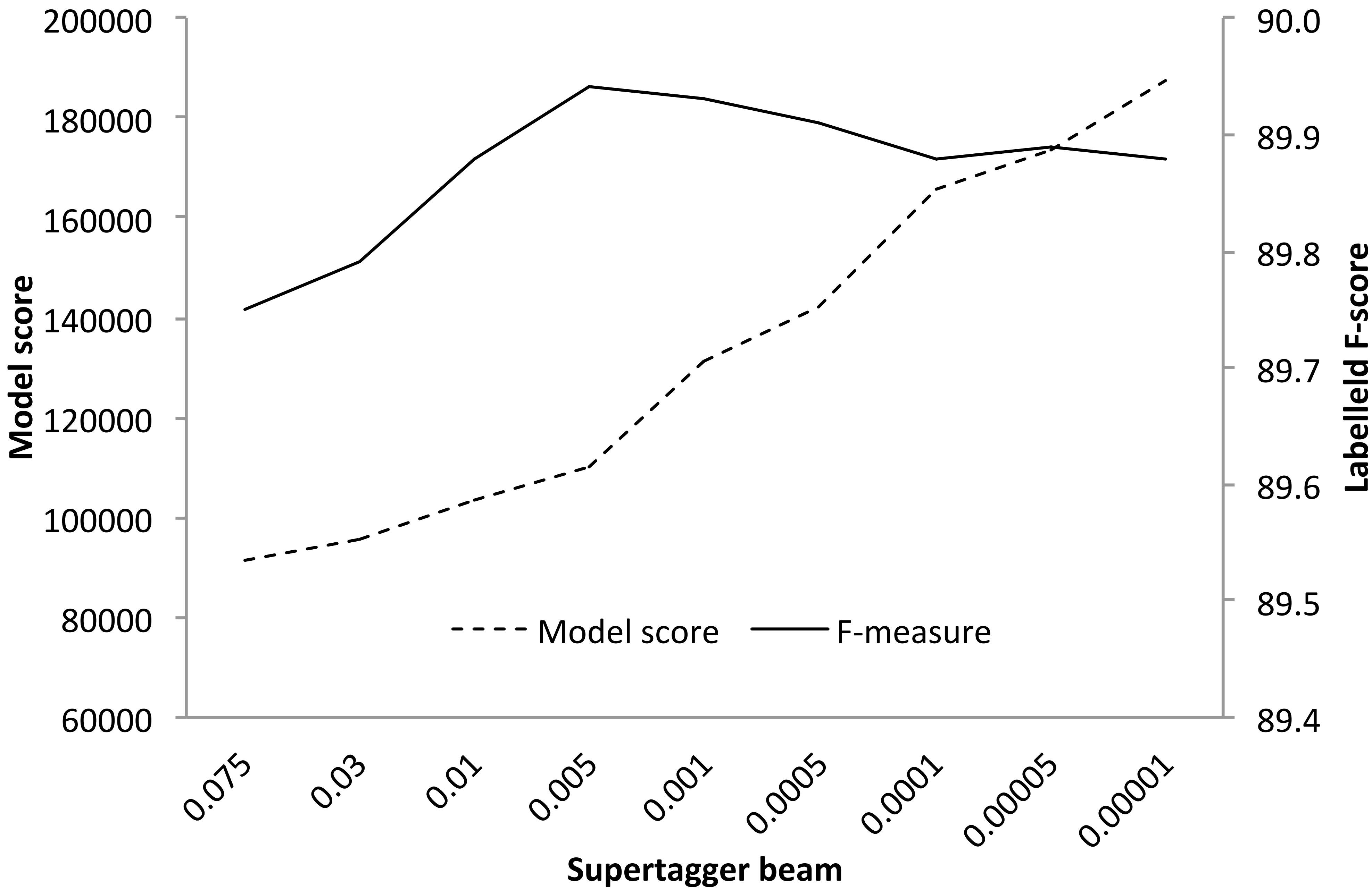
- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65
- Belief propagation (1 iter):
  - tight beam: 88.19
  - loose beam: **88.80**
- Dual decomposition (25 iter):
  - tight beam: 88.14
  - loose beam: **88.80**

Experiments

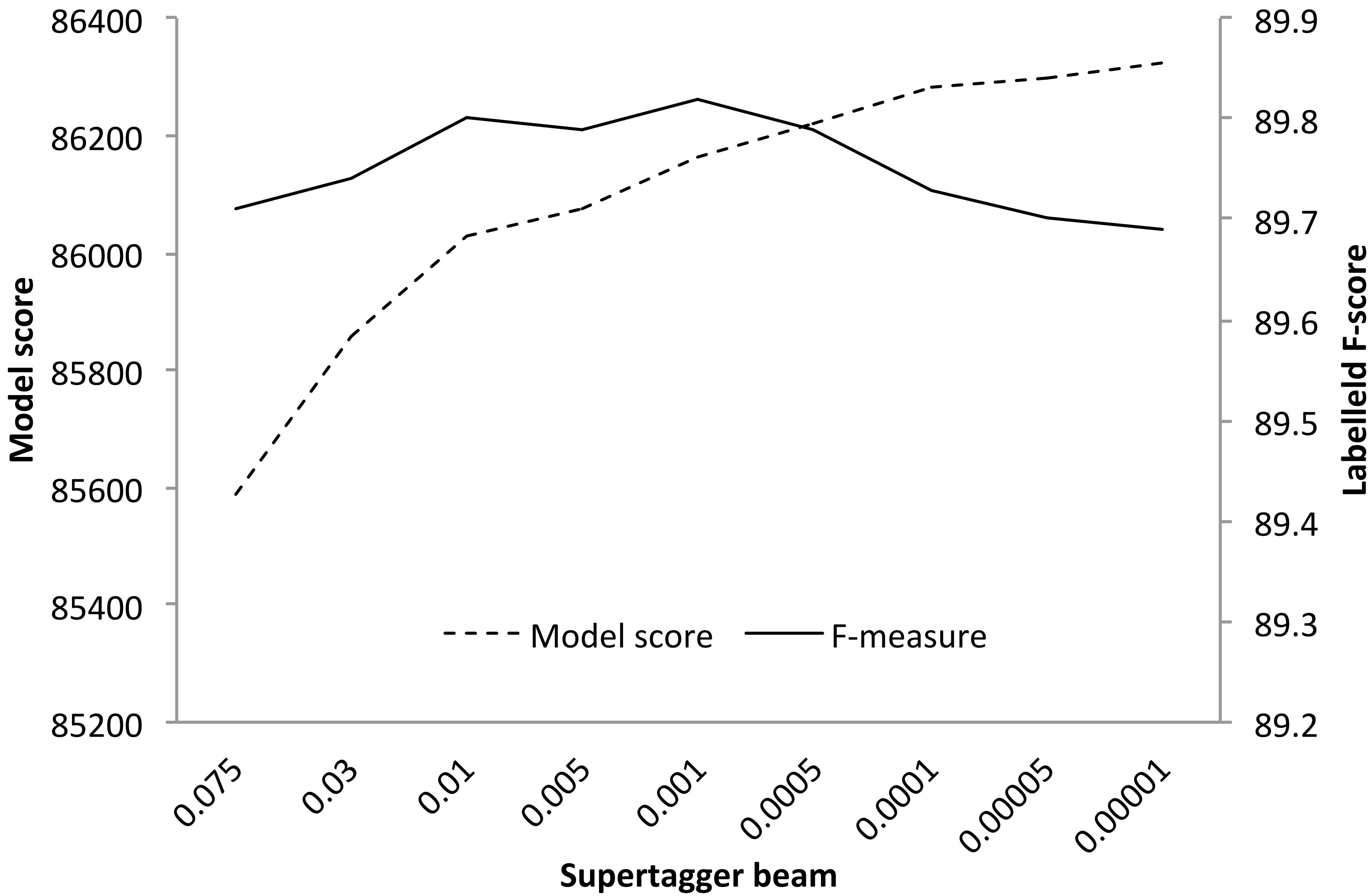
- Baseline results on test:
  - tight beam: 87.73
  - loose beam: 87.65
- Belief propagation (1 iter):
  - tight beam: 88.19
  - loose beam: **88.80**
- Dual decomposition (25 iter):
  - tight beam: 88.14
  - loose beam: **88.80**

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

- Part I: Search in Lexicalized Grammar Parsing
  Pruning and Optimality
- Part II: More Accurate Search with Combined Models
  with Loopy Belief Propagation and Dual Decomposition (Auli & Lopez 2011)
- Part III: Task-specific Optimization
  with Softmax-Margin using Exact and Approximate Loss Functions

Log-Linear Parsing Models

Log-Linear Parsing Models

- Conditional Random Fields (CRF; Lafferty et al. 2001).
  e.g. Clark & Curran (2007) and Finkel et al. (2008)

Log-Linear Parsing Models

- Conditional Random Fields (CRF; Lafferty et al. 2001).
  e.g. Clark & Curran (2007) and Finkel et al. (2008)
- Maximizing conditional log-likelihood (CLL).

Log-Linear Parsing Models

- Conditional Random Fields (CRF; Lafferty et al. 2001).
  e.g. Clark & Curran (2007) and Finkel et al. (2008)
- Maximizing conditional log-likelihood (CLL).
- Optimizing for task-specific metrics leads to better performance (Goodman, 1996; Och, 2003).

Log-Linear Parsing Models

- Conditional Random Fields (CRF; Lafferty et al. 2001).
  e.g. Clark & Curran (2007) and Finkel et al. (2008)
- Maximizing conditional log-likelihood (CLL).
- Optimizing for task-specific metrics leads to better performance (Goodman, 1996; Och, 2003).
- Softmax-Margin (SMM) objective (Sha & Saul, 2006; Povey & Woodland, 2008; Gimpel & Smith, 2010).

Softmax-Margin

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

weights features input

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

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

weights features input true output proposed output

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

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

- Re-weights outcomes by *risk.*
- Risk is the *loss* incurred.
- Loss function an *unweighted feature* -- if **decomposable.**

Losses for Parsing

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

$T(y)$ = 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 $t \in T(y)$

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

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

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Losses for Parsing

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

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

Losses are ***decomposable*** -- only use information within local sub-structure (Taskar et al., 2004)

Labelled, directed dependency recovery
(Clark & Hockenmaier, 2002)

Approximate Losses for Parsing

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

Losses are ***decomposable*** -- only use information within local sub-structure (Taskar et al., 2004)

Labelled, directed dependency recovery
(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

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

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

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

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

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

Metrics do not decompose over parses

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

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

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

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

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

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

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

! Metrics do not decompose over parses

... but statistics do!

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Exact Loss Functions for Parsing

y = dependencies in ground truth

y' = dependencies in proposed output

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

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

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

! Metrics do not decompose over parses

... but statistics do!

Use state-split dynamic program to compute F_1-augmented expectations for losses on sentence-level!

Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)

Standard CKY

items $A_{i,j}$

Standard CKY

items $A_{i,j}$

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

Standard CKY

items $A_{i,j}$

target analysis

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

Standard CKY

items $A_{i,j}$

target analysis
correct dependencies
dependencies

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

Standard CKY

items $A_{i,j}$

target analysis
correct dependencies
dependencies

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

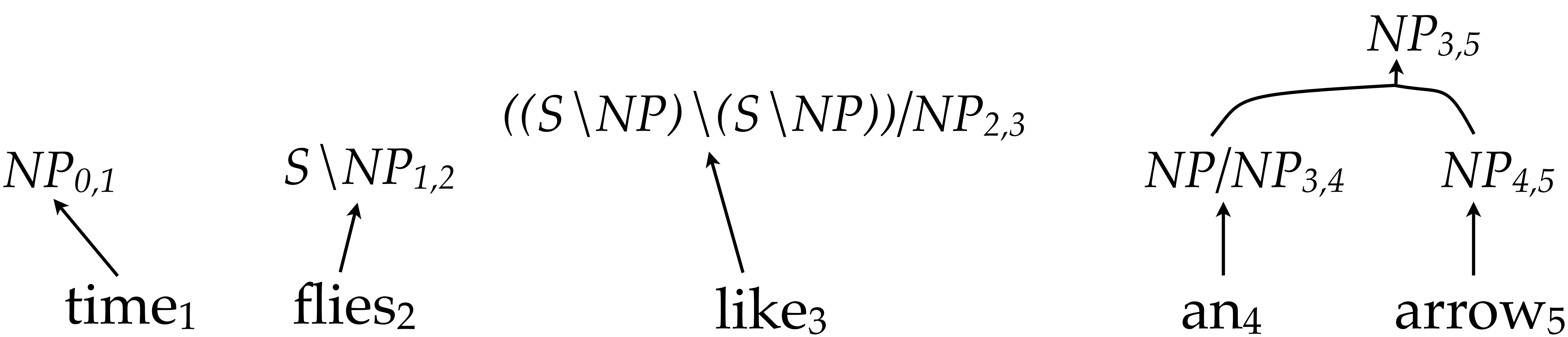
$NP_{0,1}$ $S\backslash NP_{1,2}$ $NP/NP_{3,4}$ $NP_{4,5}$

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

Standard CKY

items $A_{i,j}$

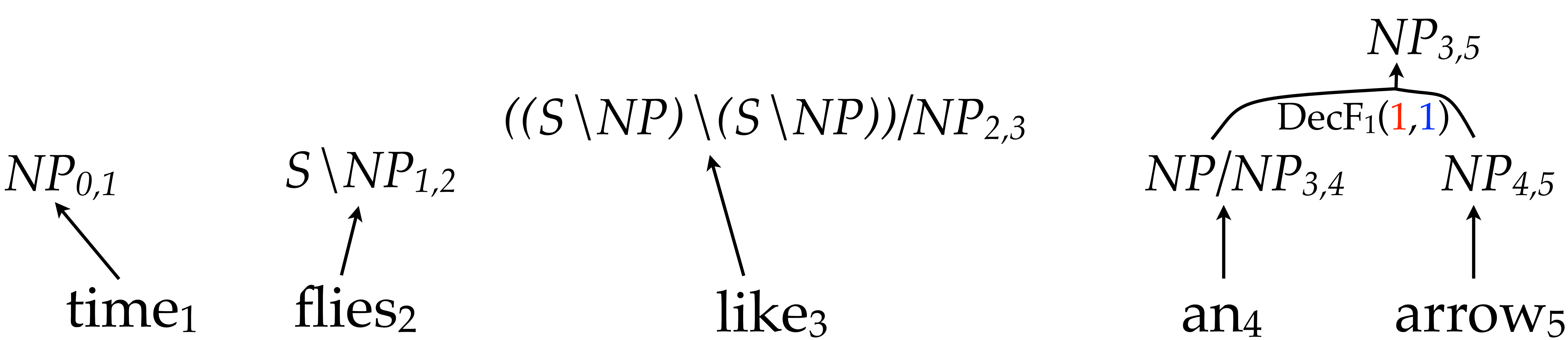
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

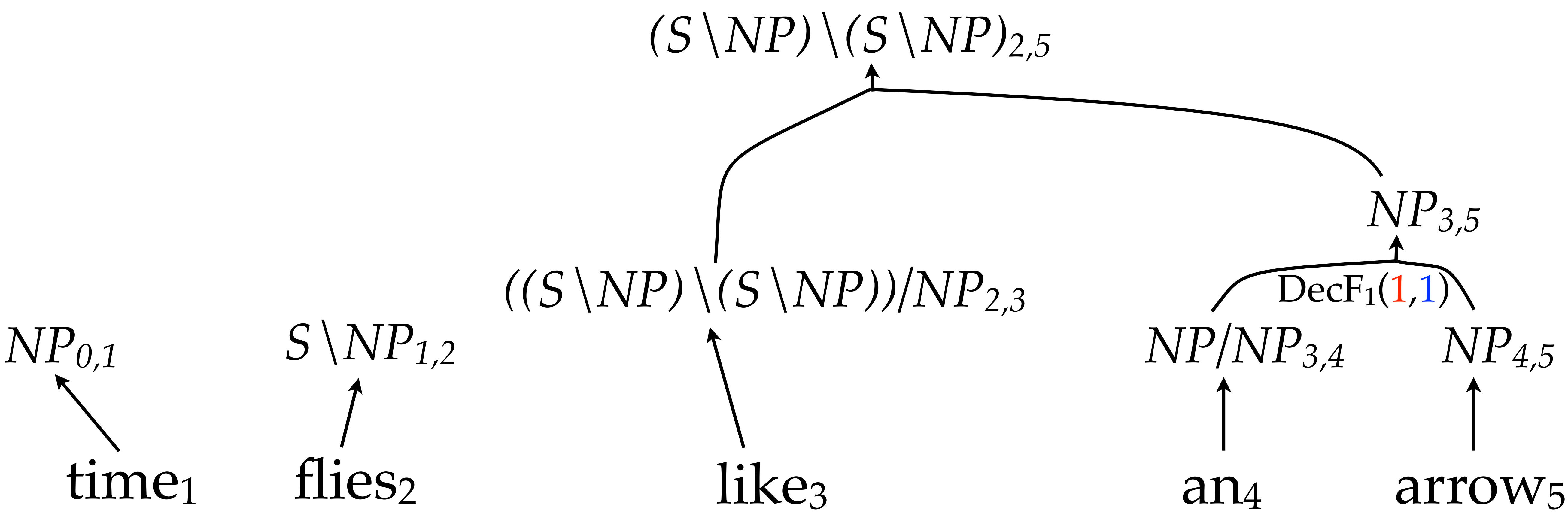
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

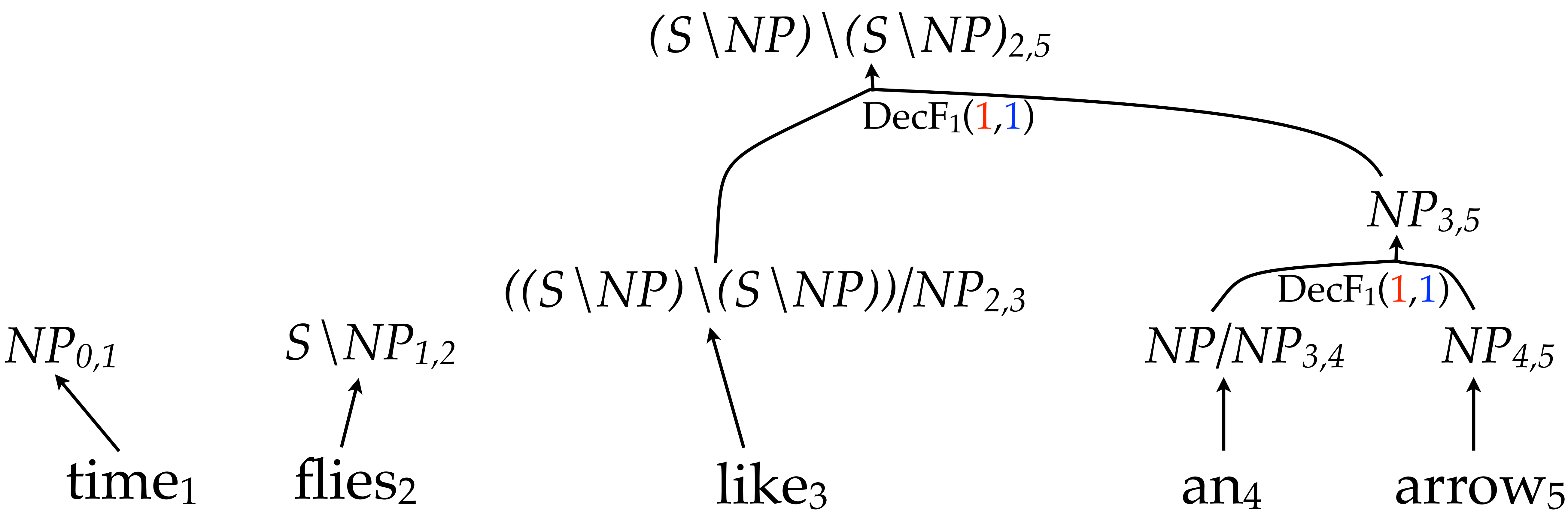
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

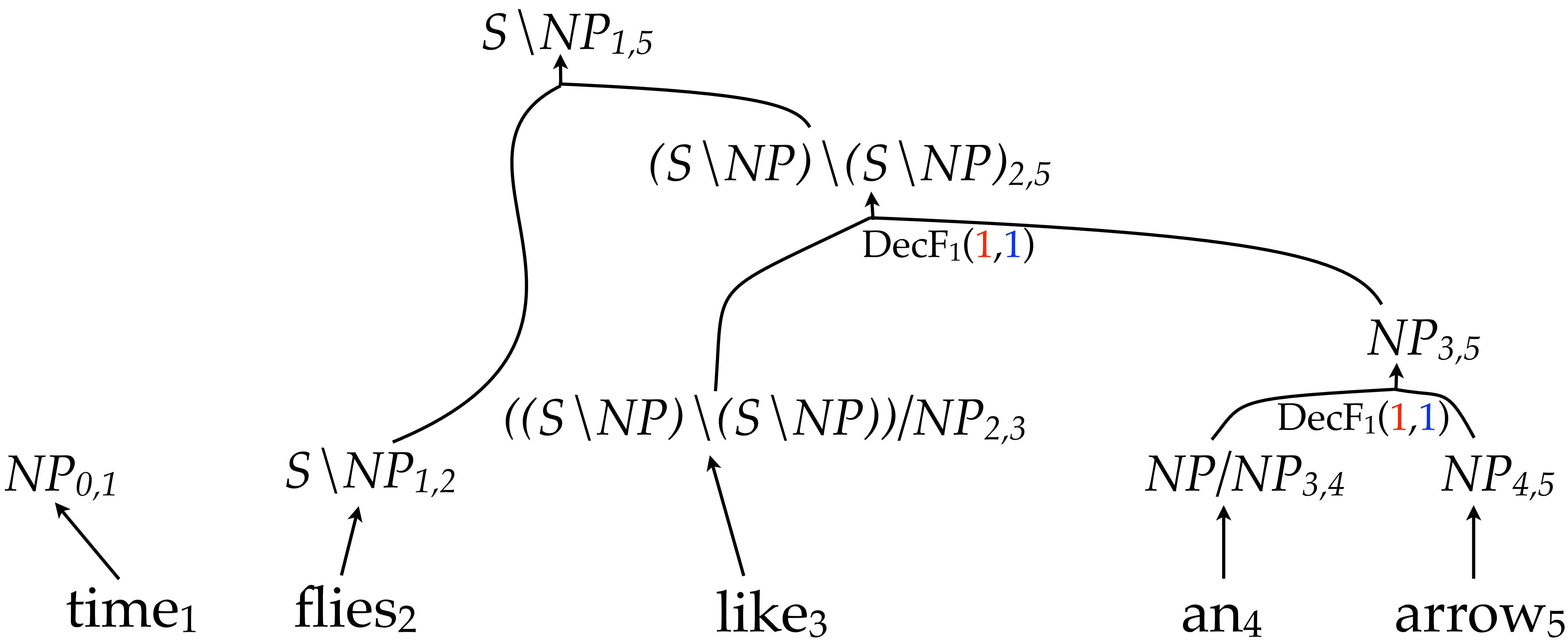
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

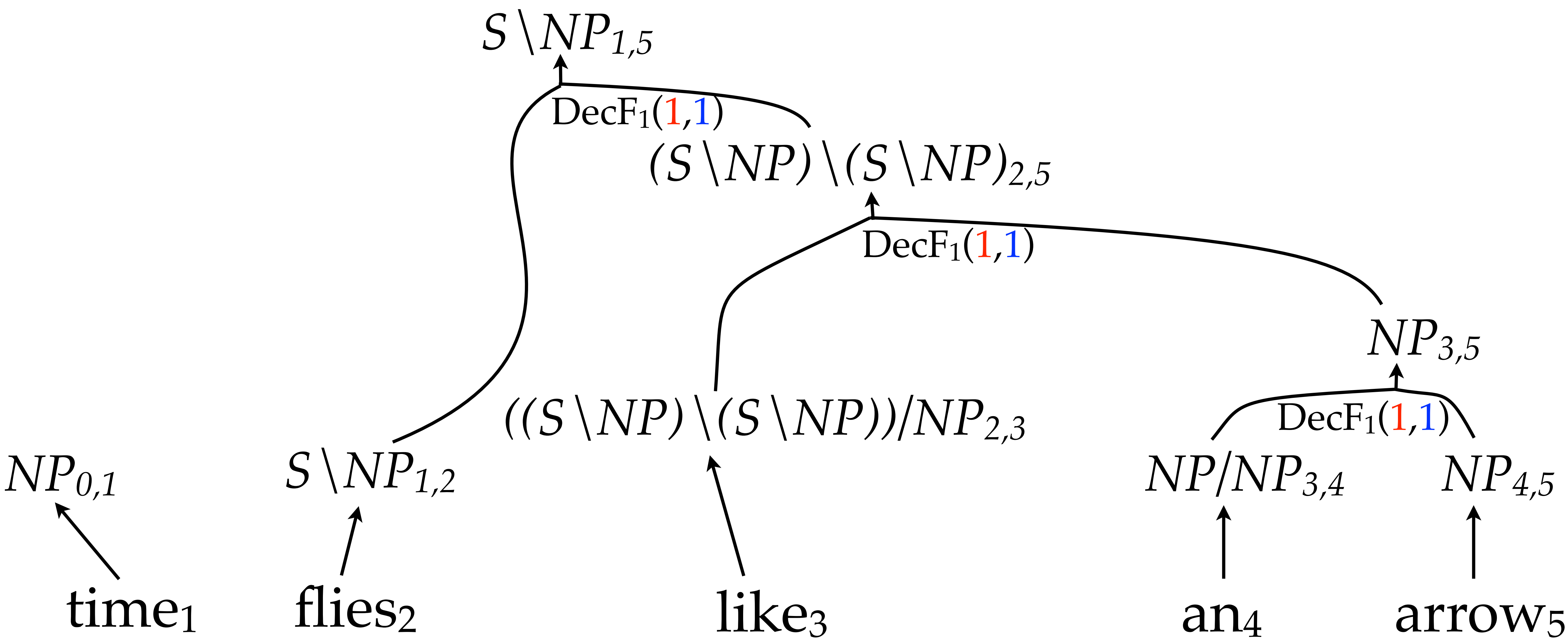
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

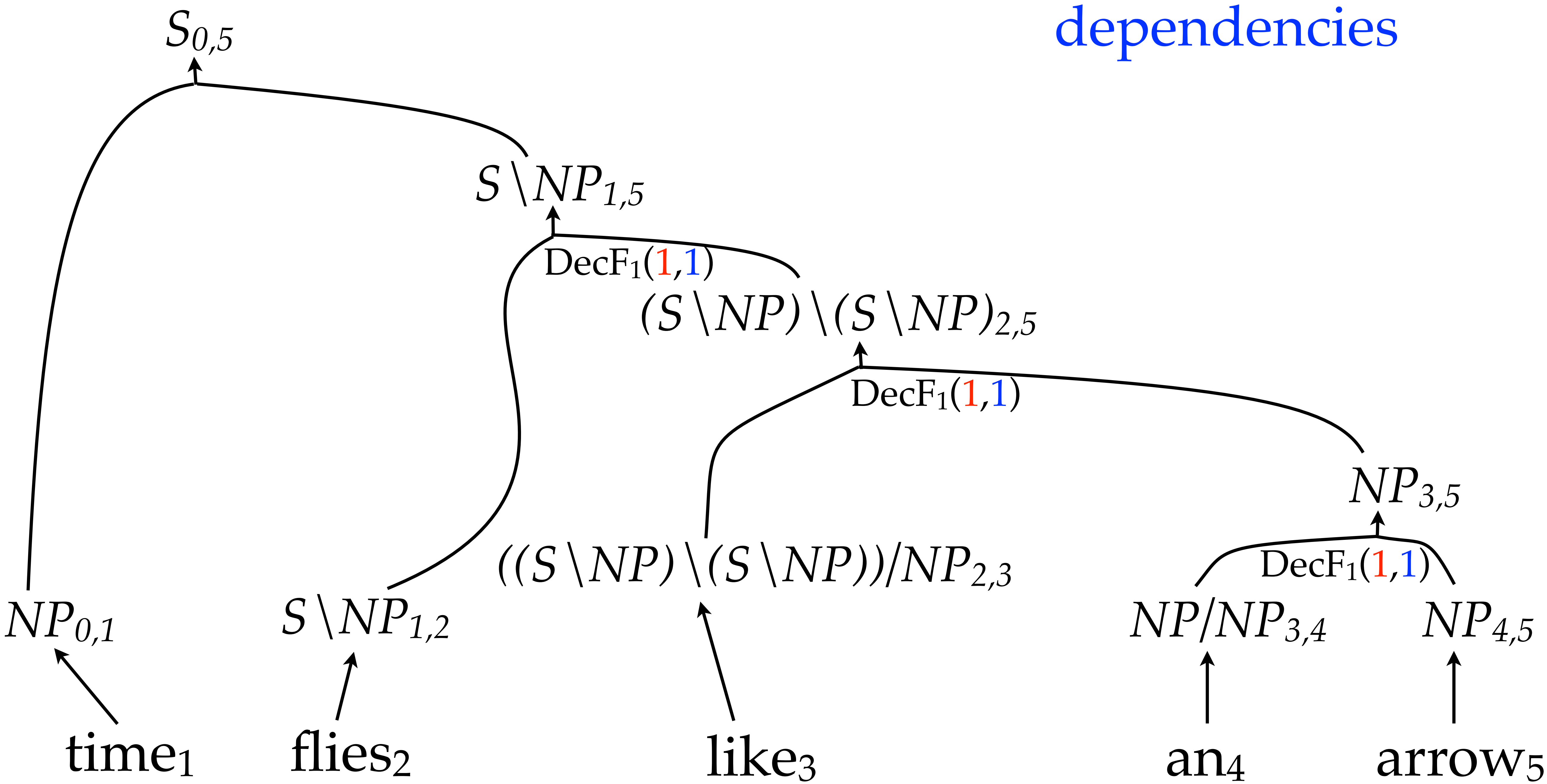
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

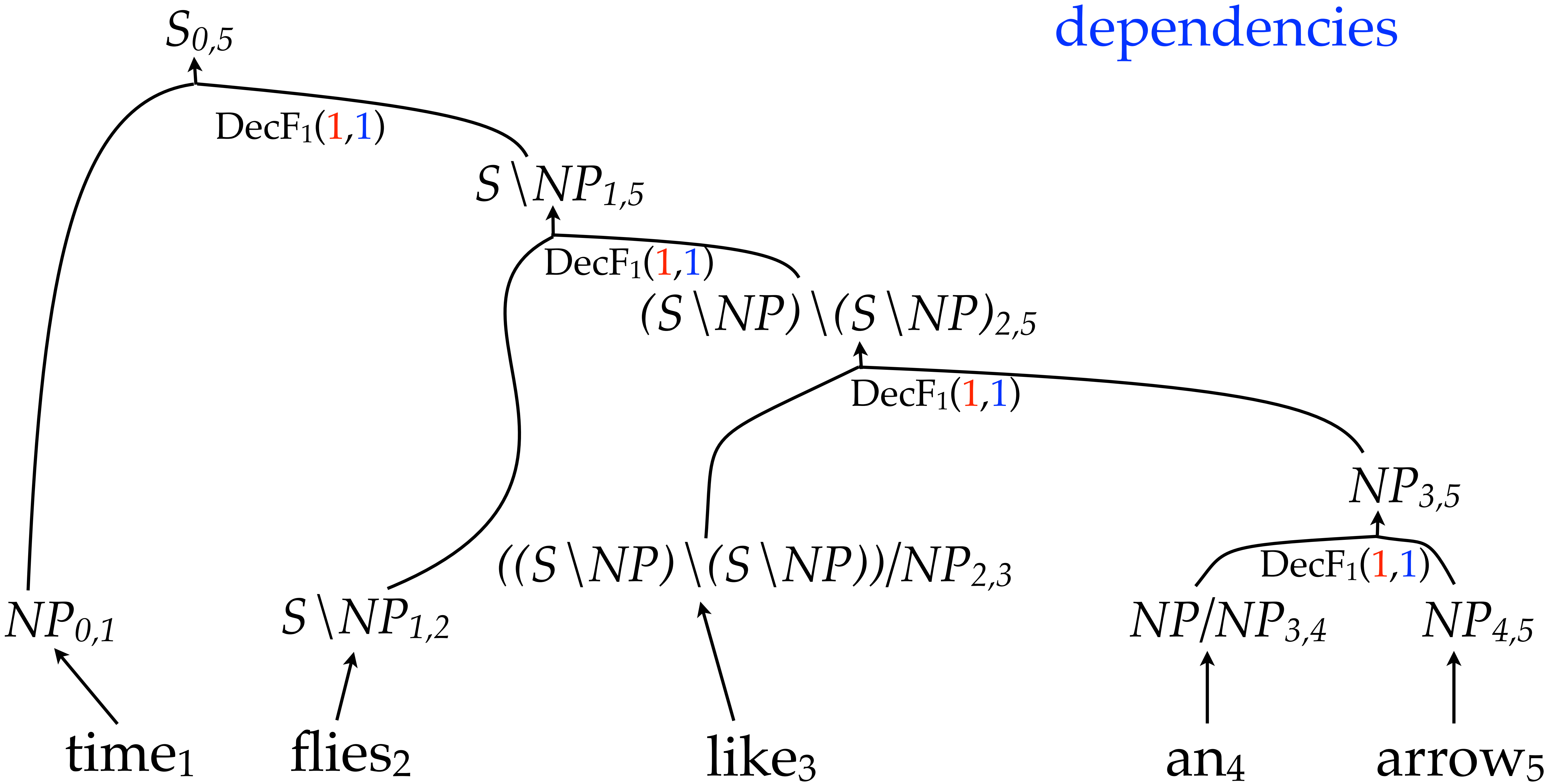
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

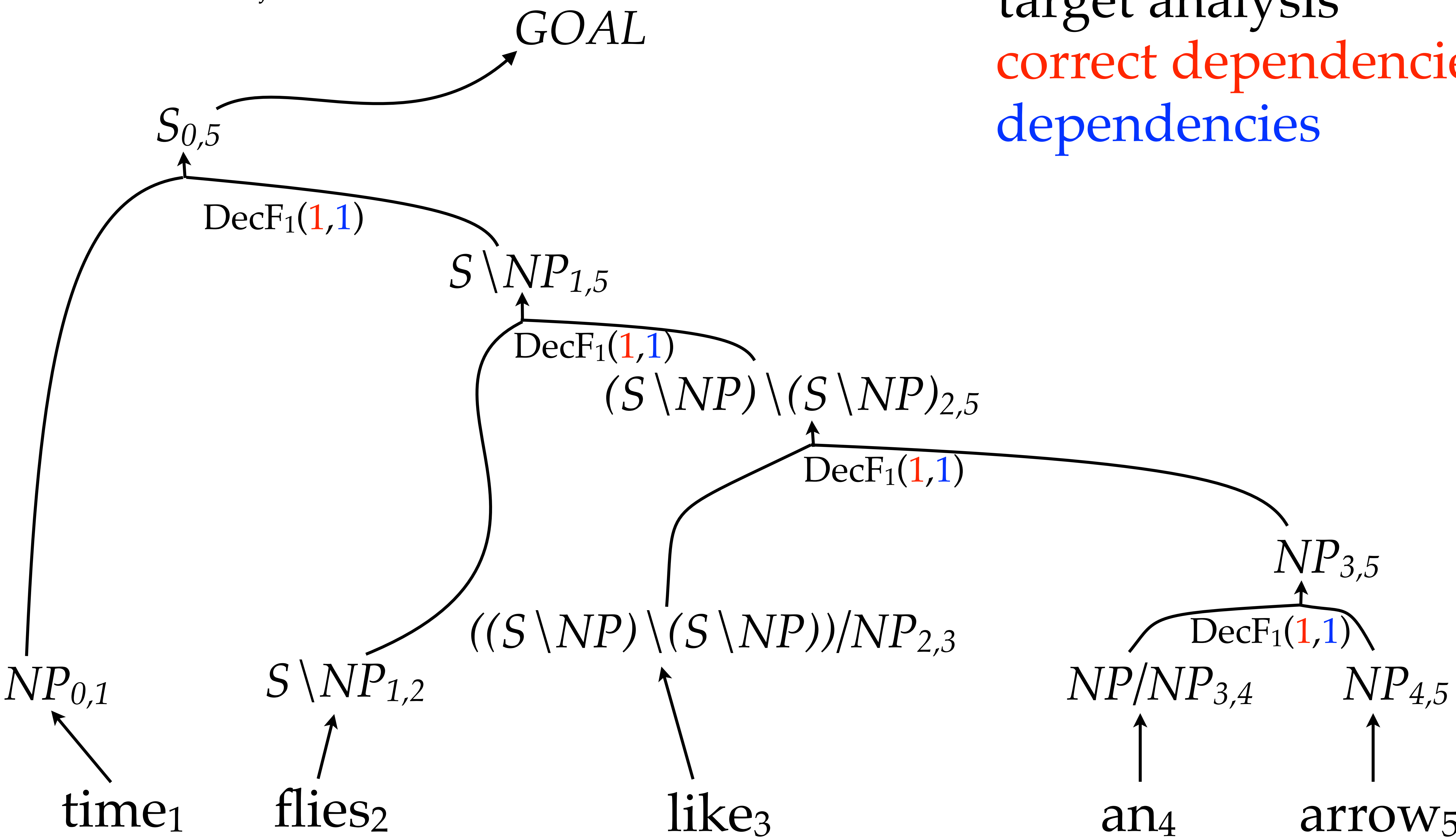
target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

target analysis
correct dependencies
dependencies



Standard CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

Standard CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

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

Standard CKY

items $A_{i,j}$

another analysis
correct dependencies
all dependencies

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

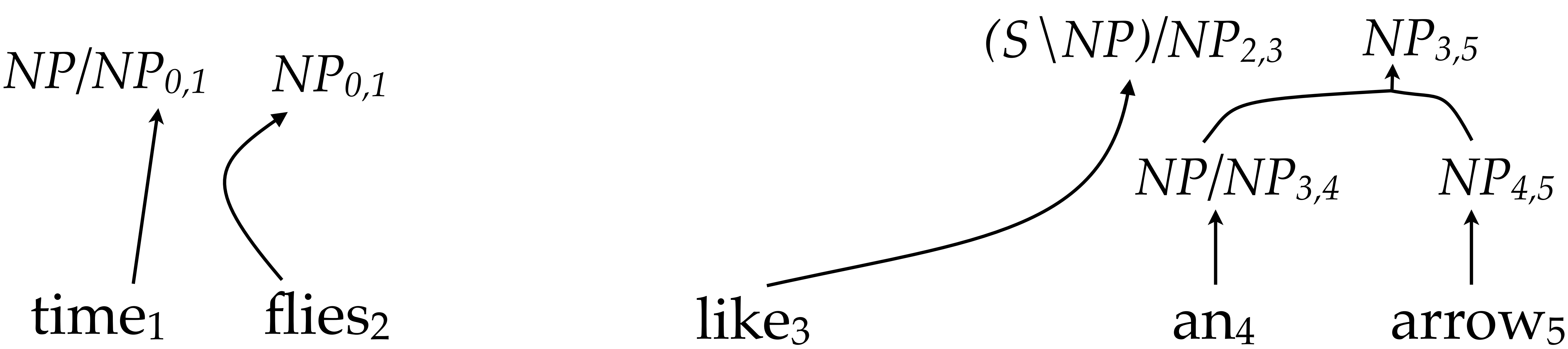
$NP/NP_{0,1}$ $NP_{0,1}$ $NP/NP_{3,4}$ $NP_{4,5}$

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

Standard CKY

items $A_{i,j}$

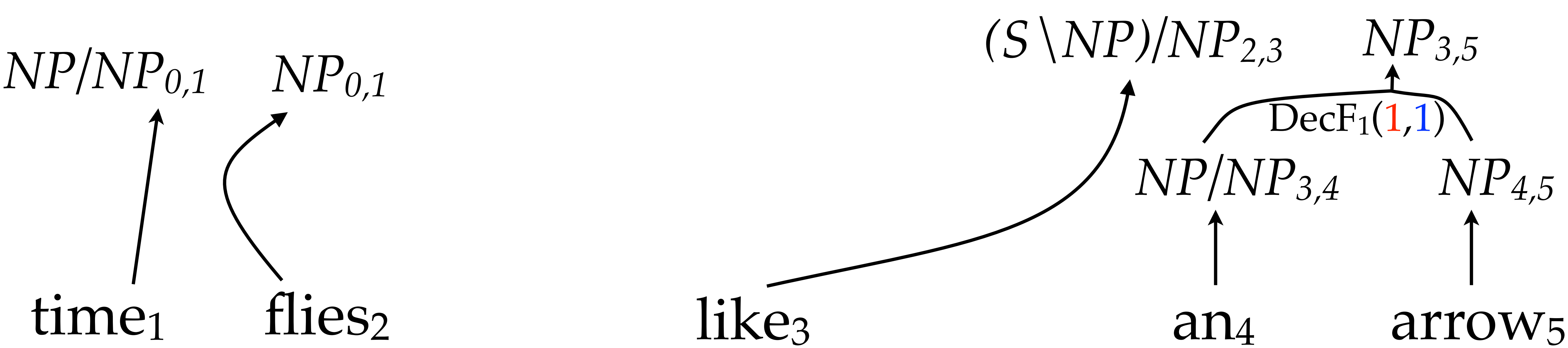
another analysis
correct dependencies
all dependencies



Standard CKY

items $A_{i,j}$

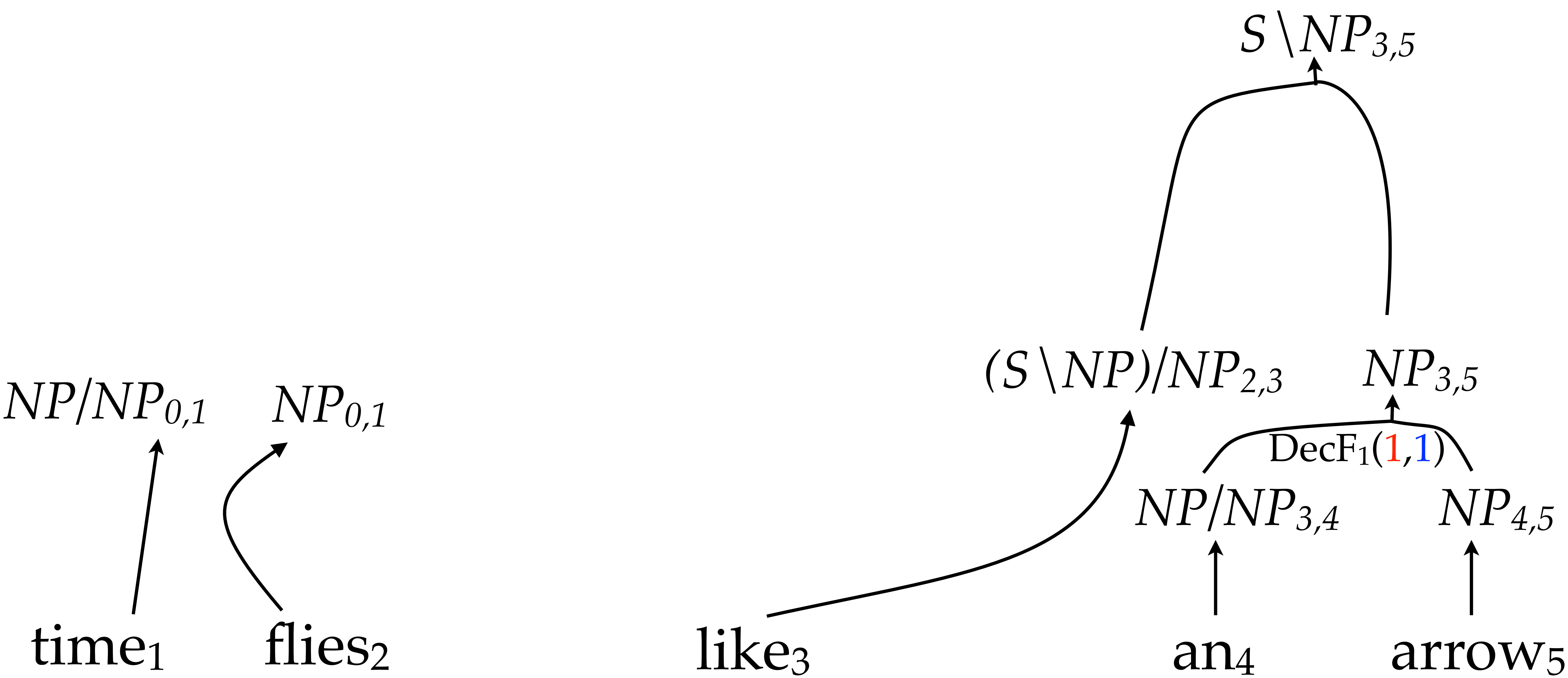
another analysis
correct dependencies
all dependencies



Standard CKY

items $A_{i,j}$

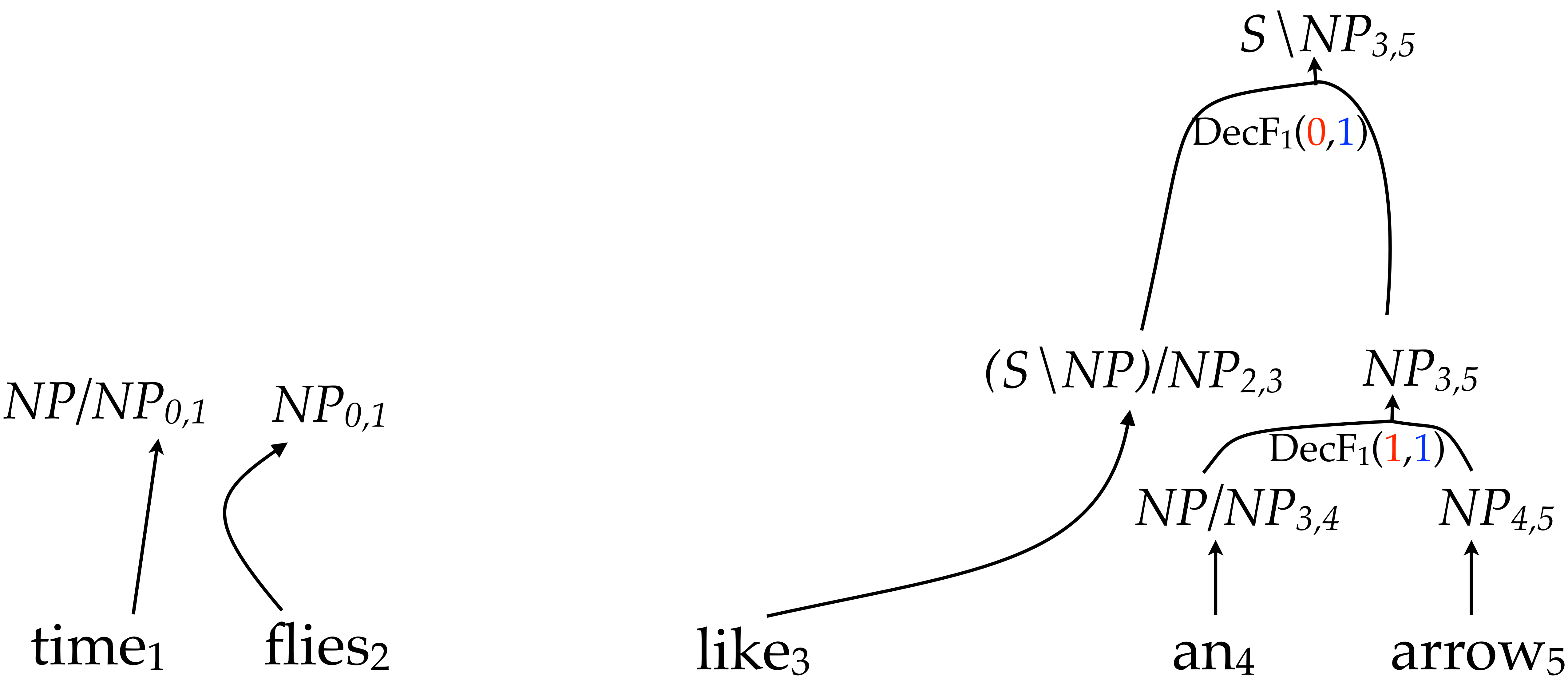
another analysis
correct dependencies
all dependencies



Standard CKY

items $A_{i,j}$

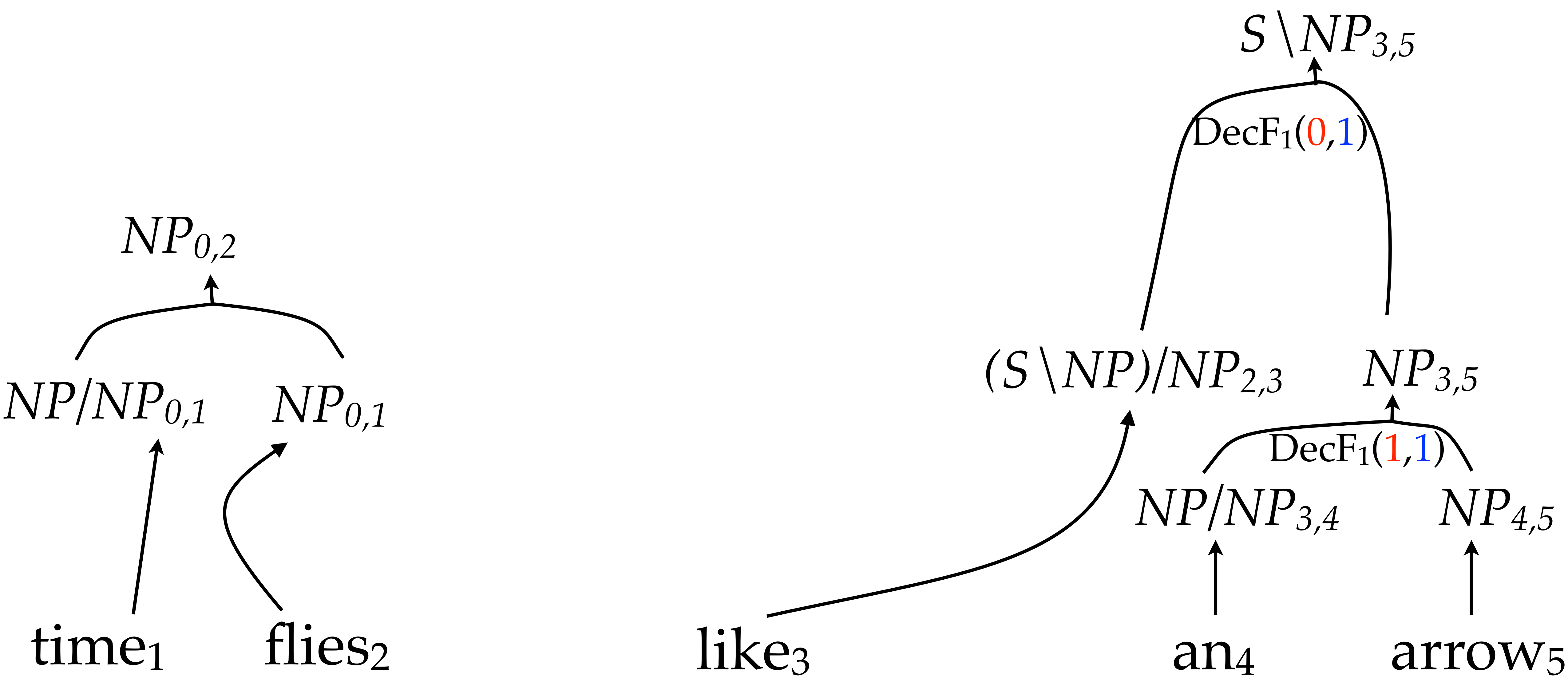
another analysis
correct dependencies
all dependencies



Standard CKY

items $A_{i,j}$

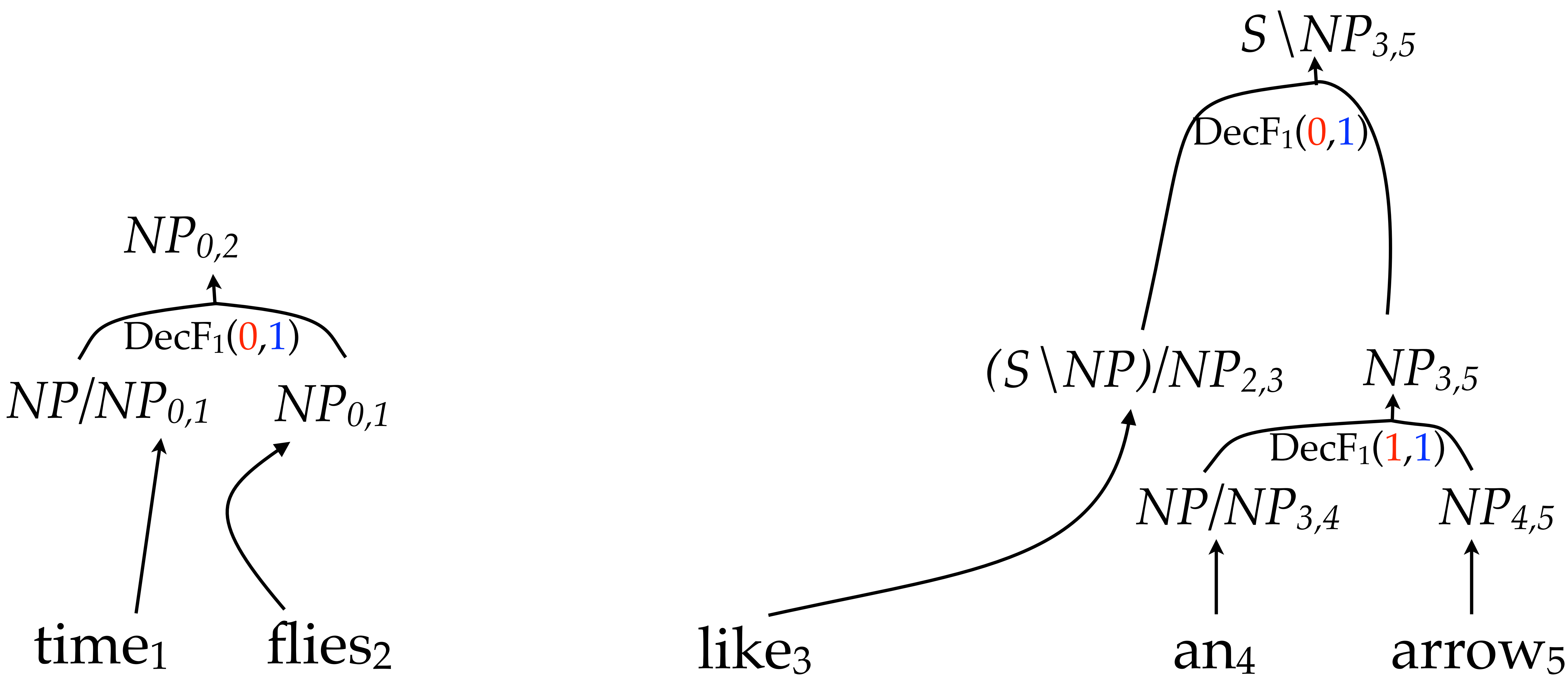
another analysis
correct dependencies
all dependencies



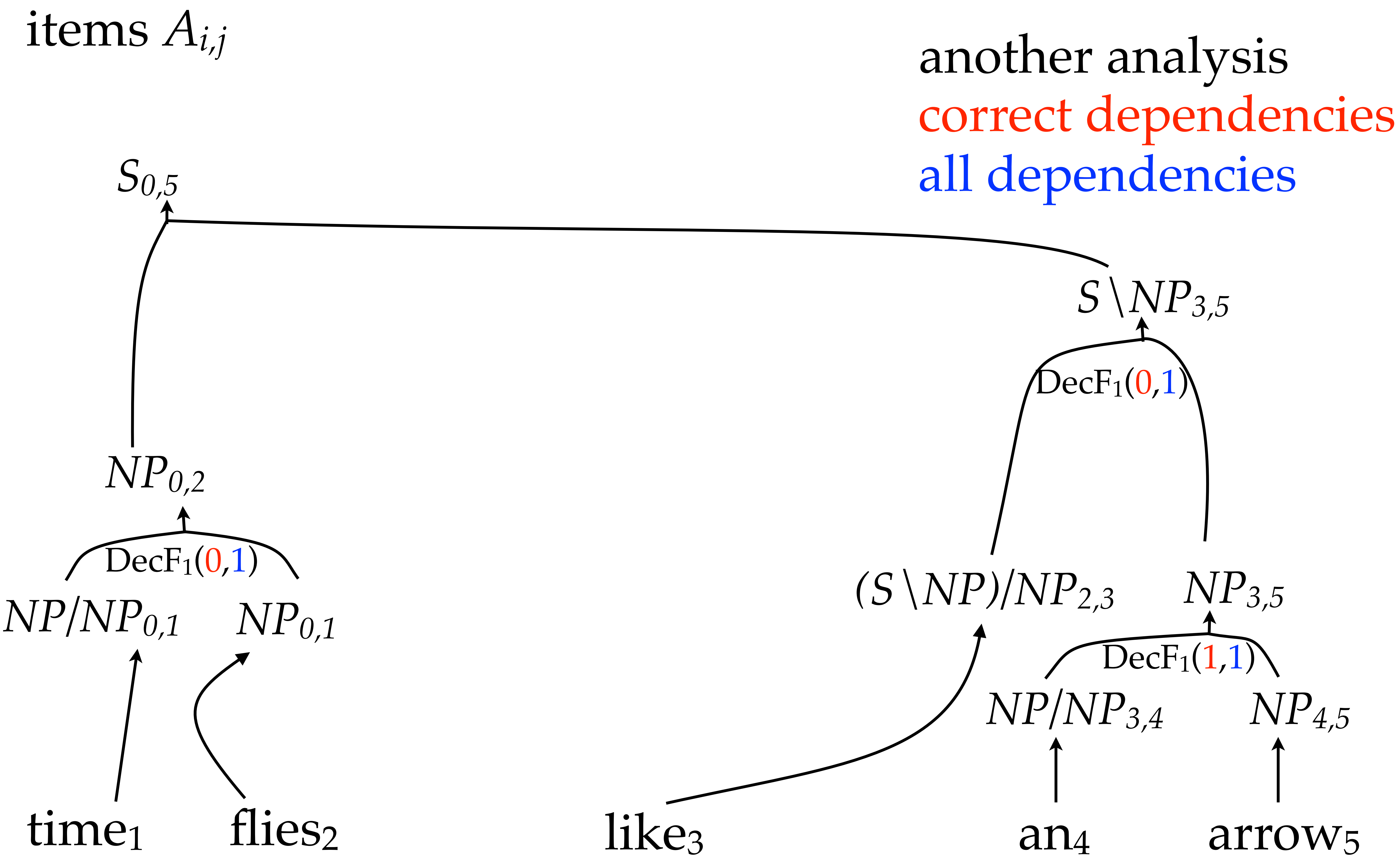
Standard CKY

items $A_{i,j}$

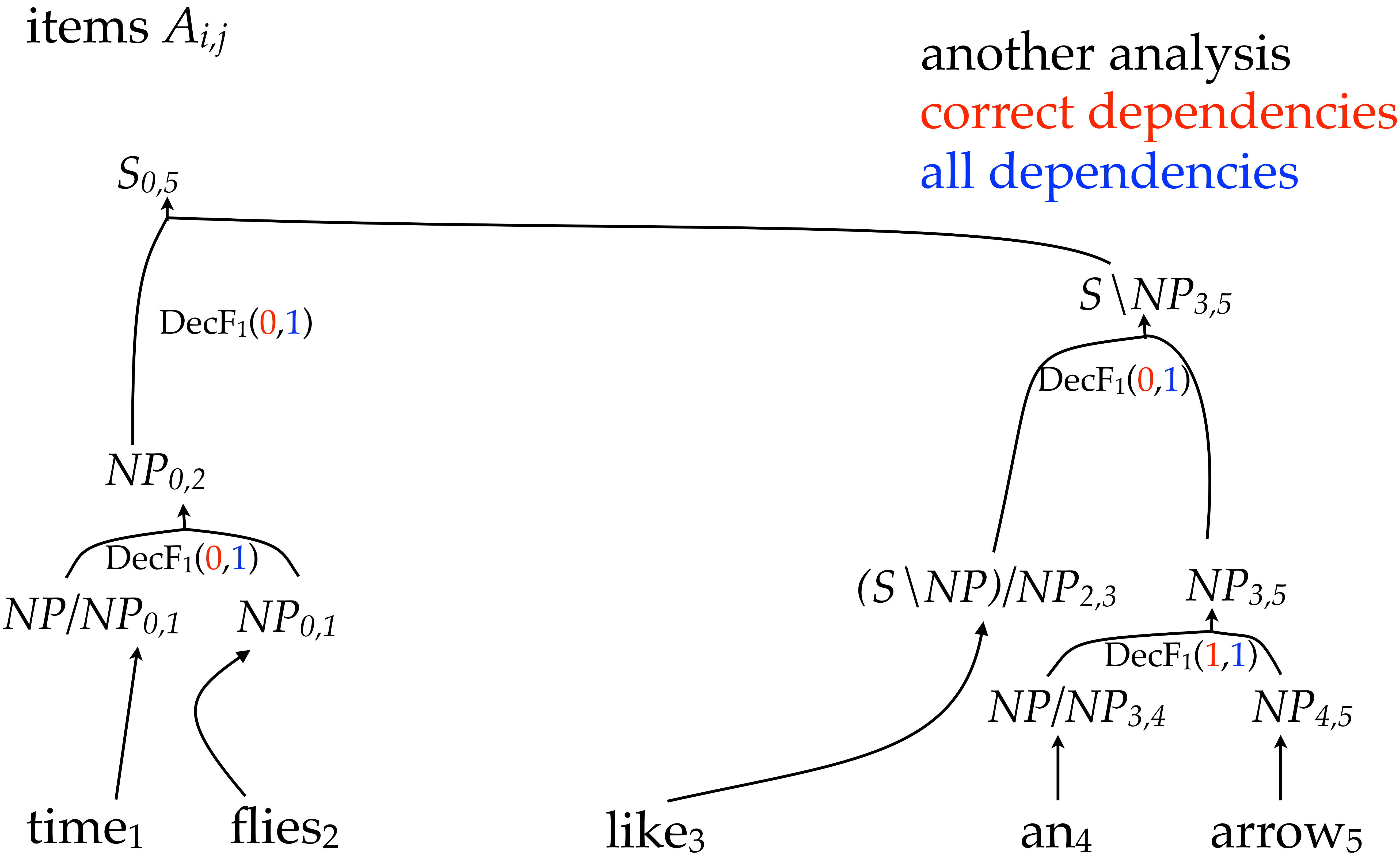
another analysis
correct dependencies
all dependencies



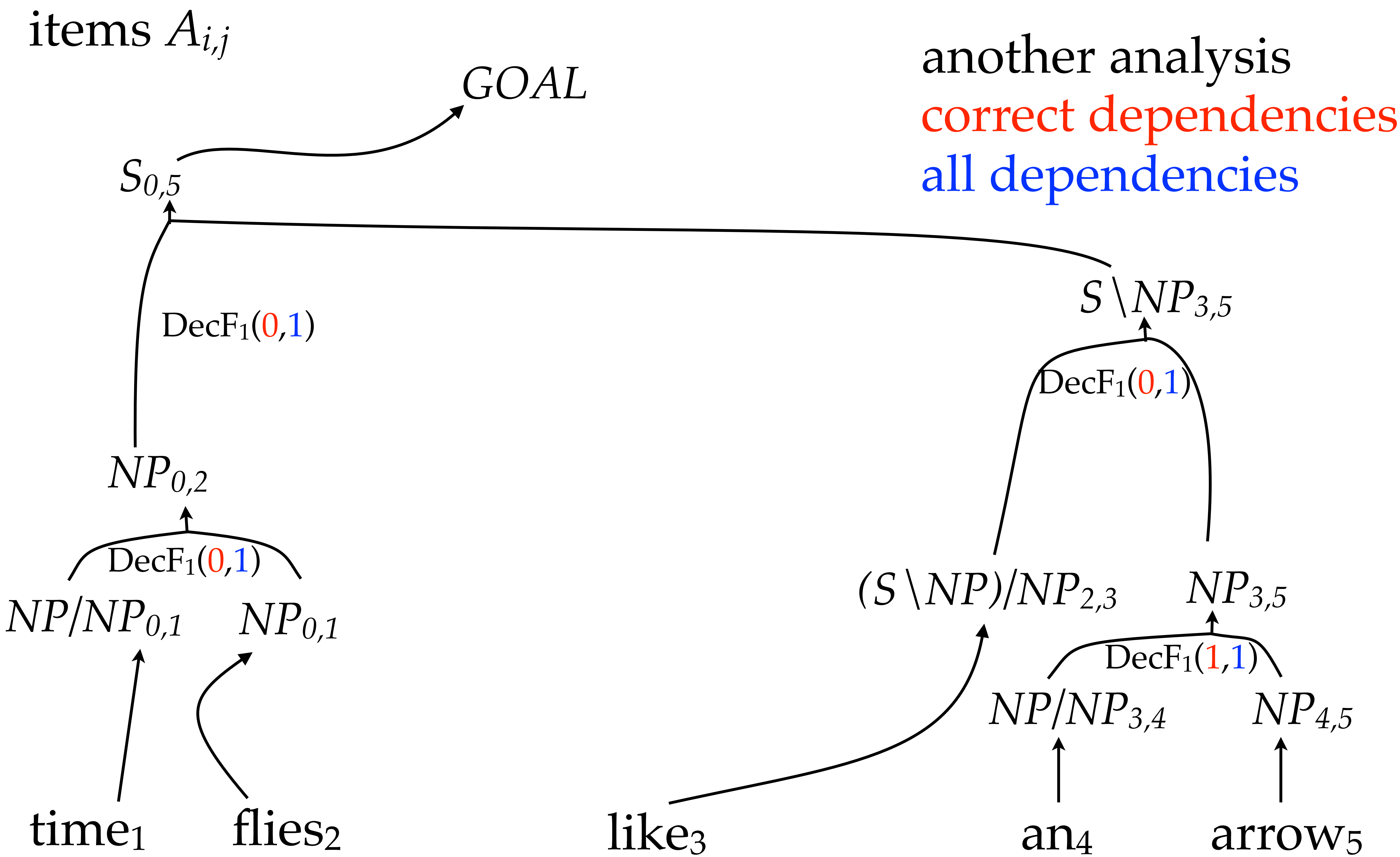
Standard CKY



Standard CKY



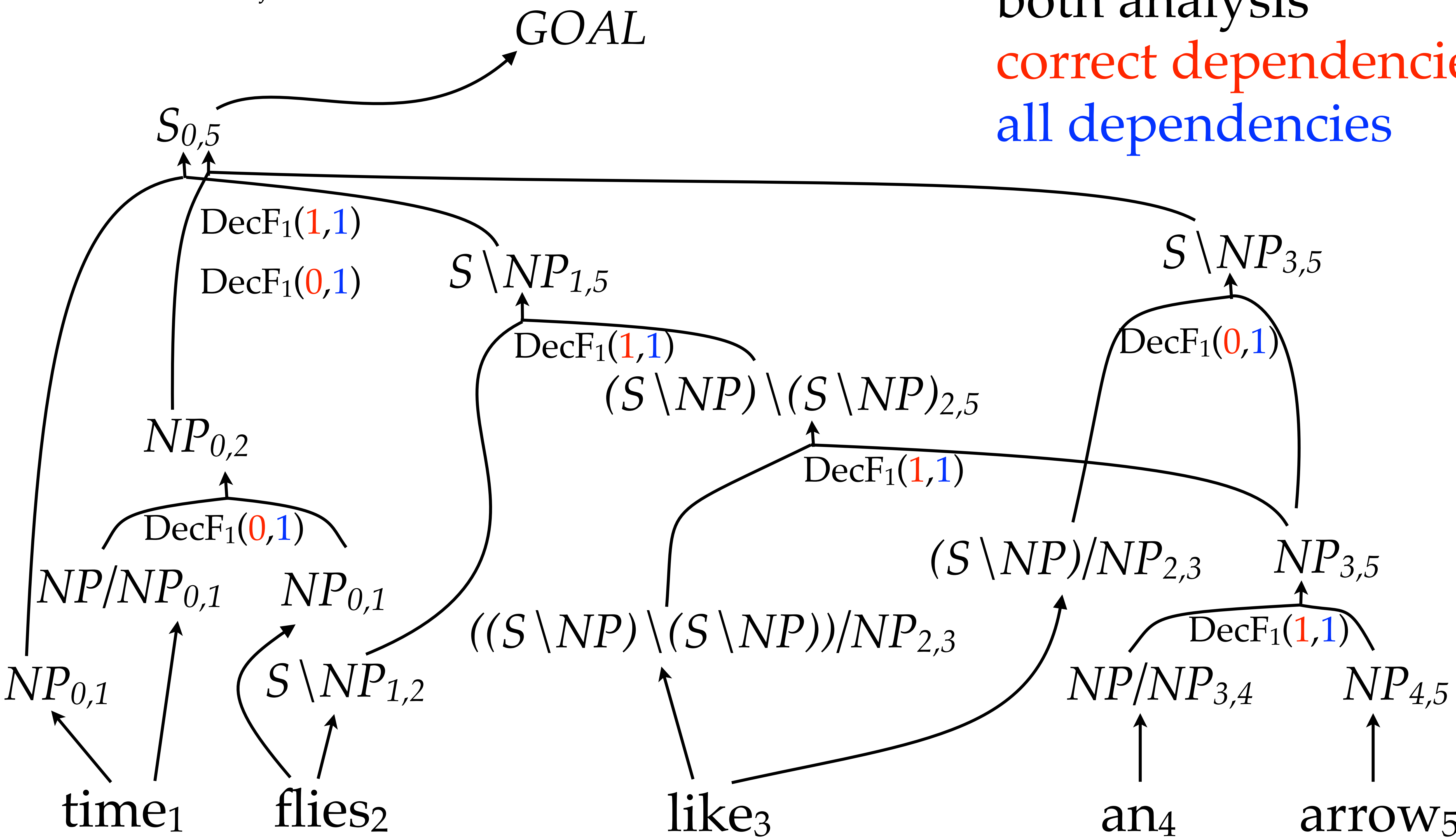
Standard CKY



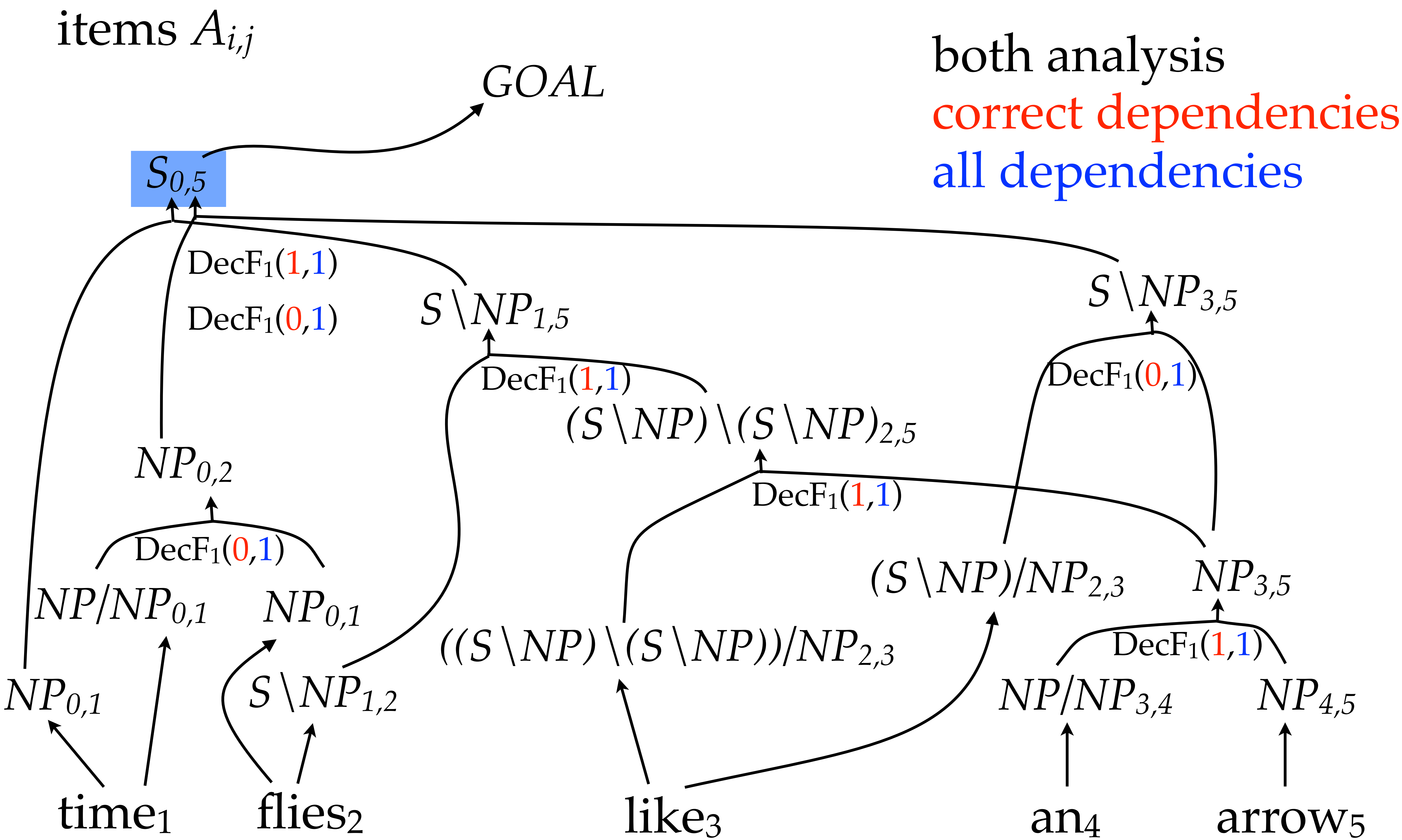
Standard CKY

items $A_{i,j}$

both analysis
correct dependencies
all dependencies



Standard CKY



State-Split CKY

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

correct dependencies

all dependencies

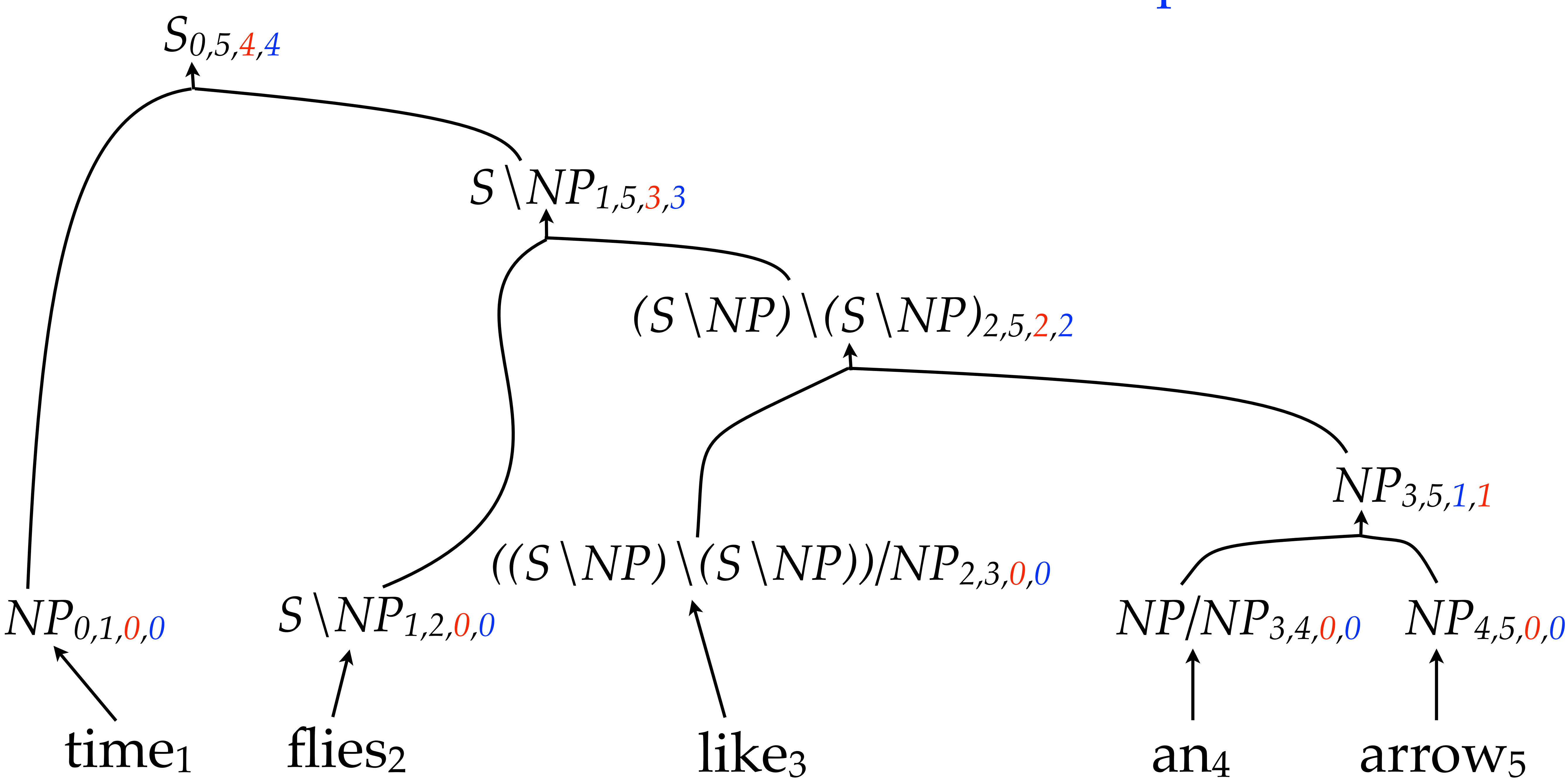
time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

State-Split CKY

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

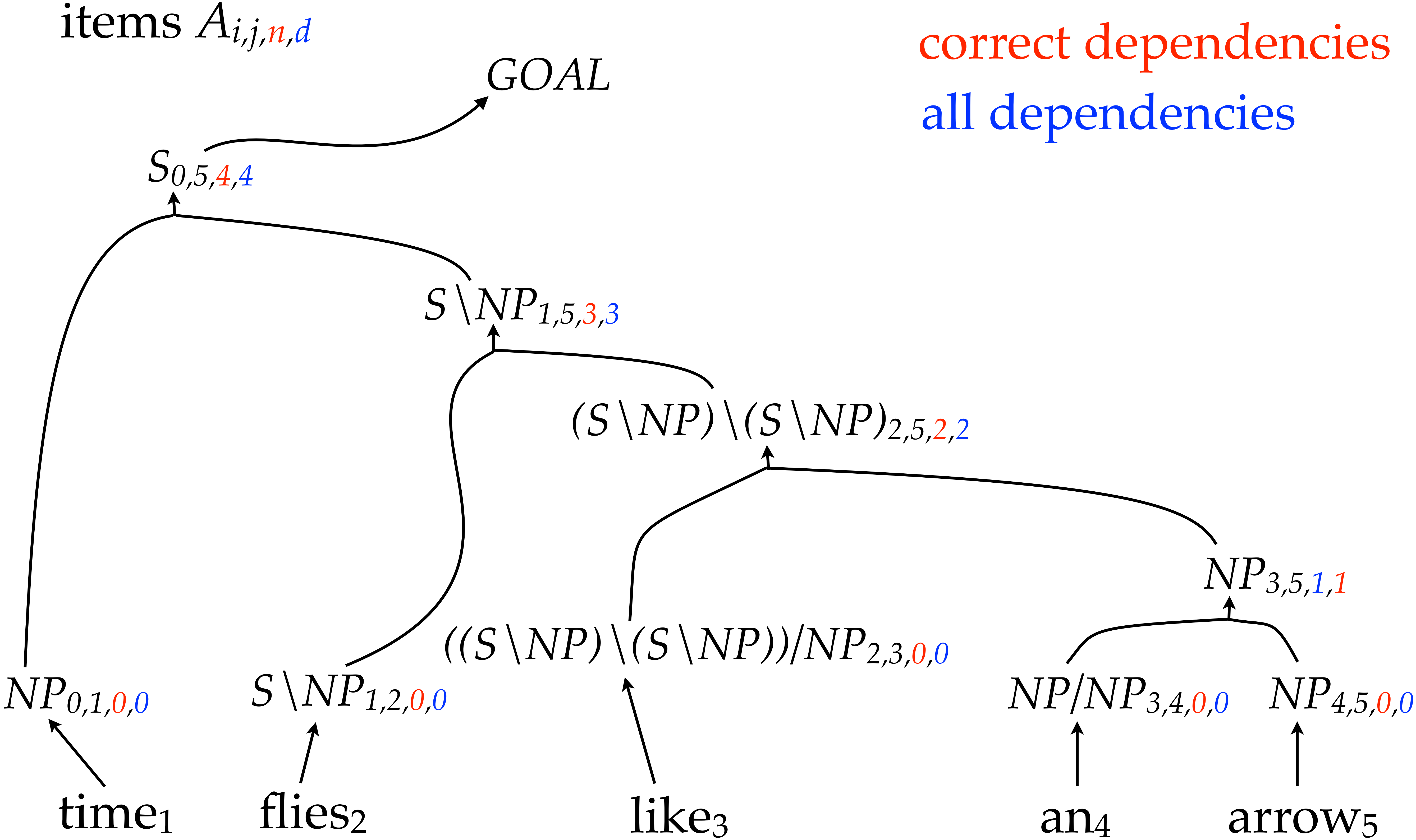
correct dependencies

all dependencies

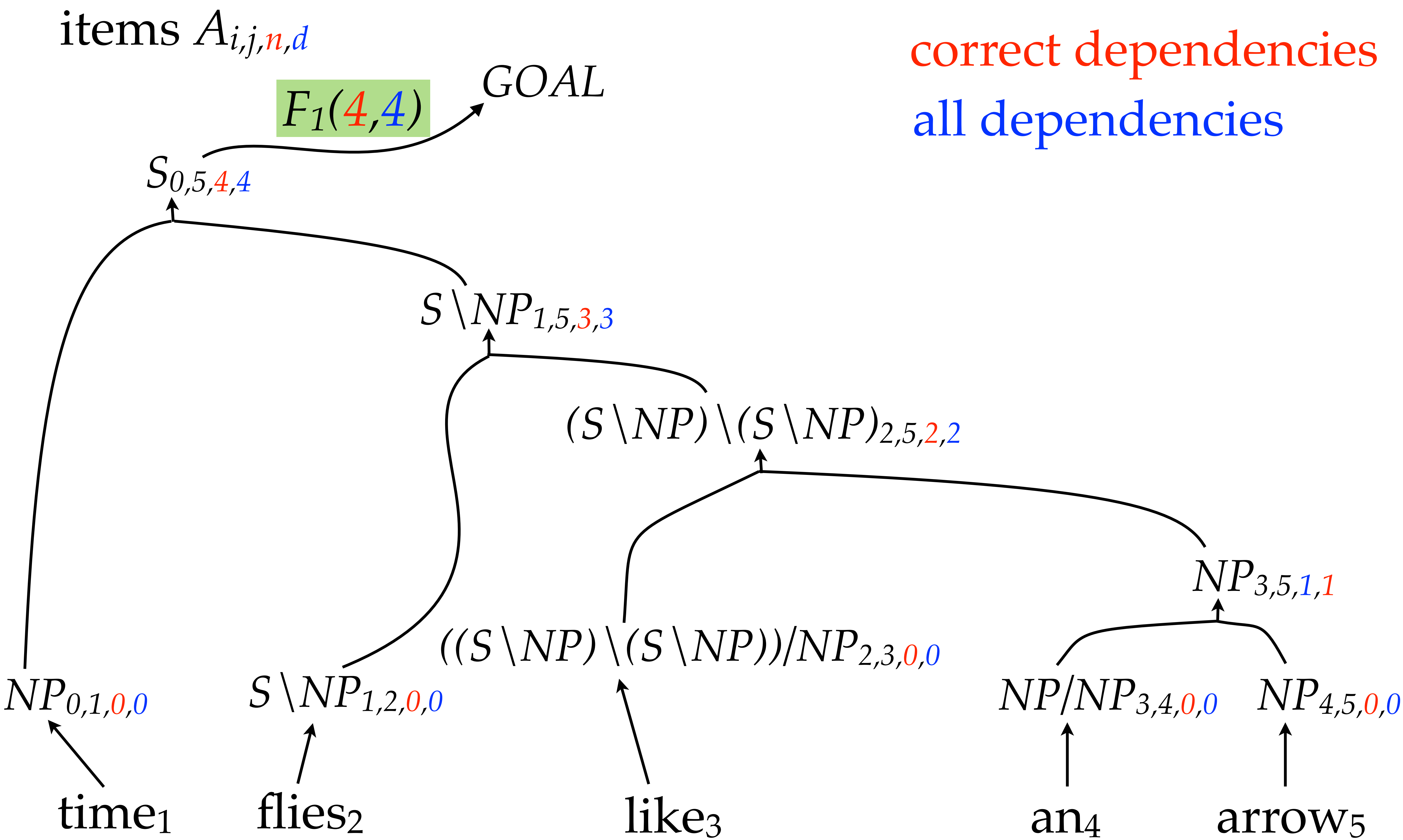


State-Split CKY

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



State-Split CKY



State-Split CKY

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

correct dependencies

all dependencies

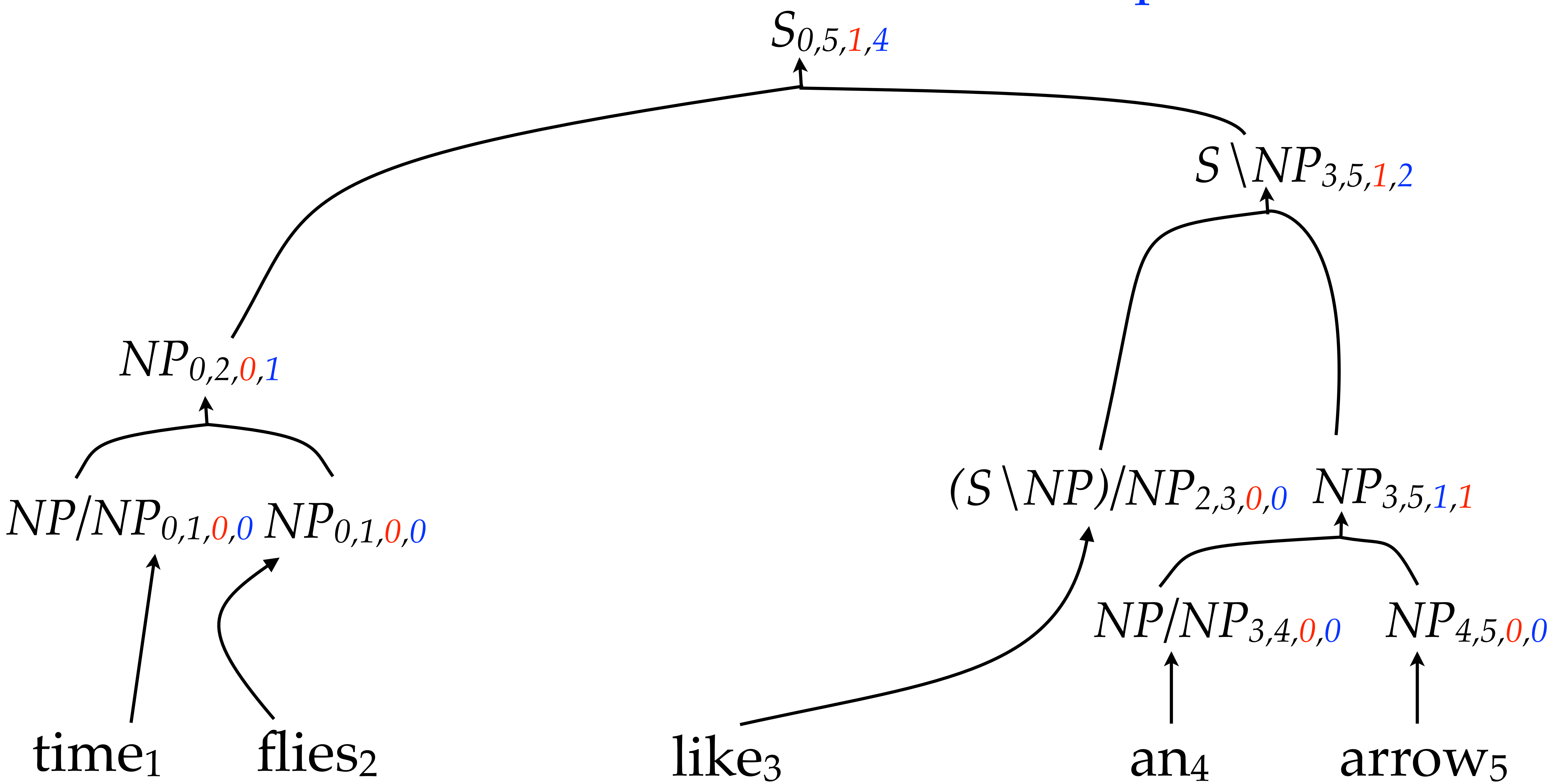
time$_1$ flies$_2$ like$_3$ an$_4$ arrow$_5$

State-Split CKY

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

correct dependencies

all dependencies

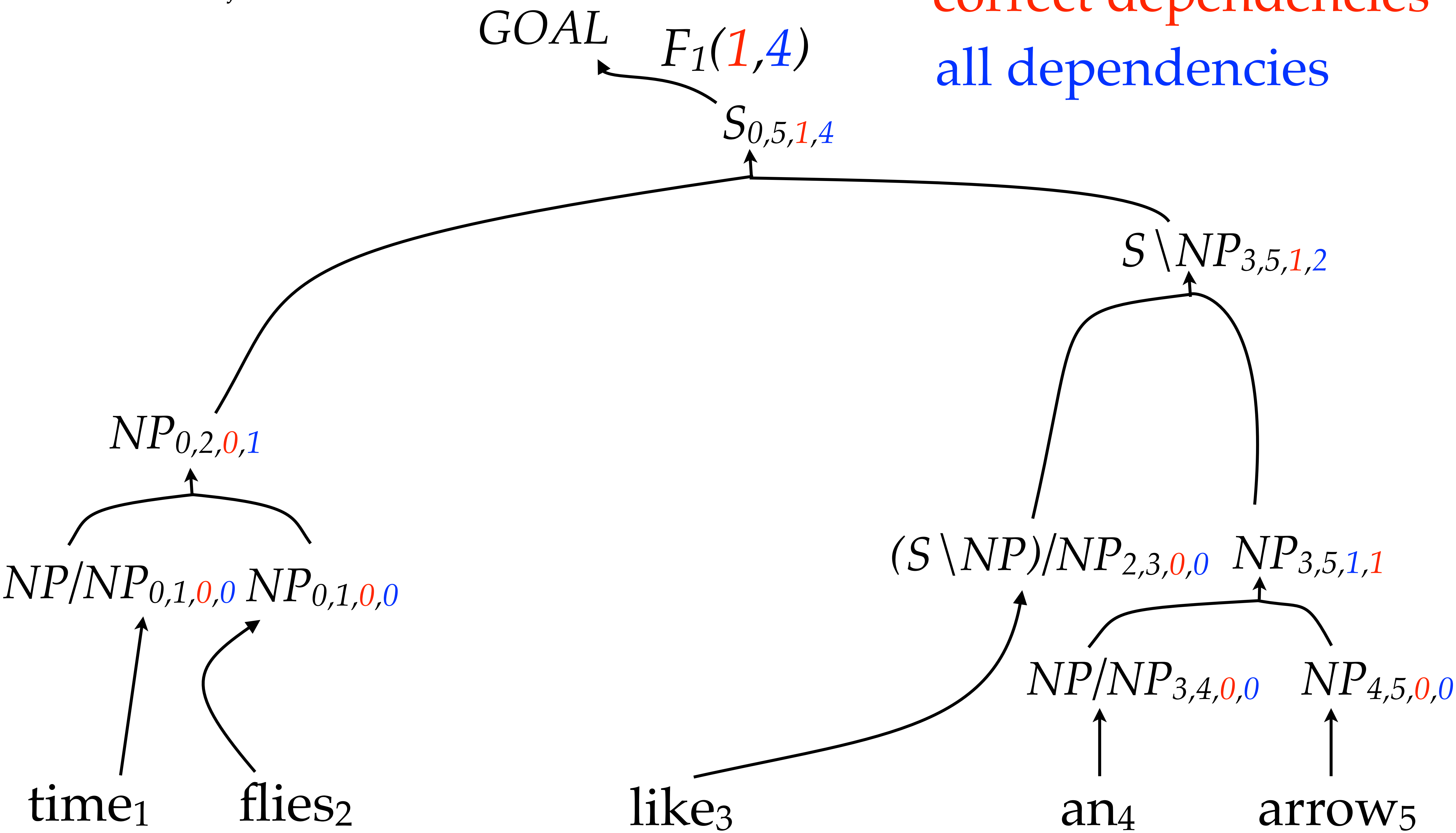


State-Split CKY

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

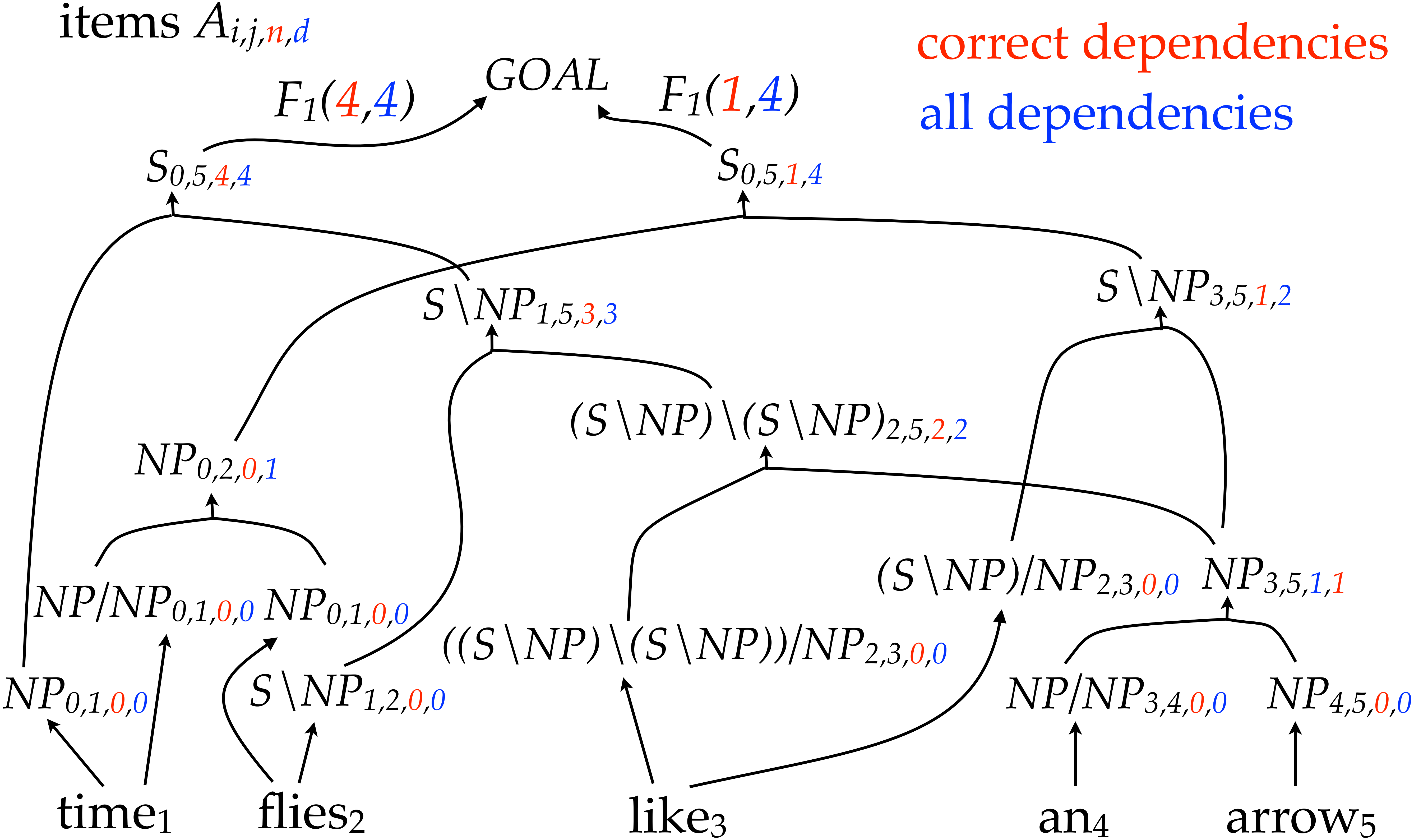
correct dependencies

all dependencies



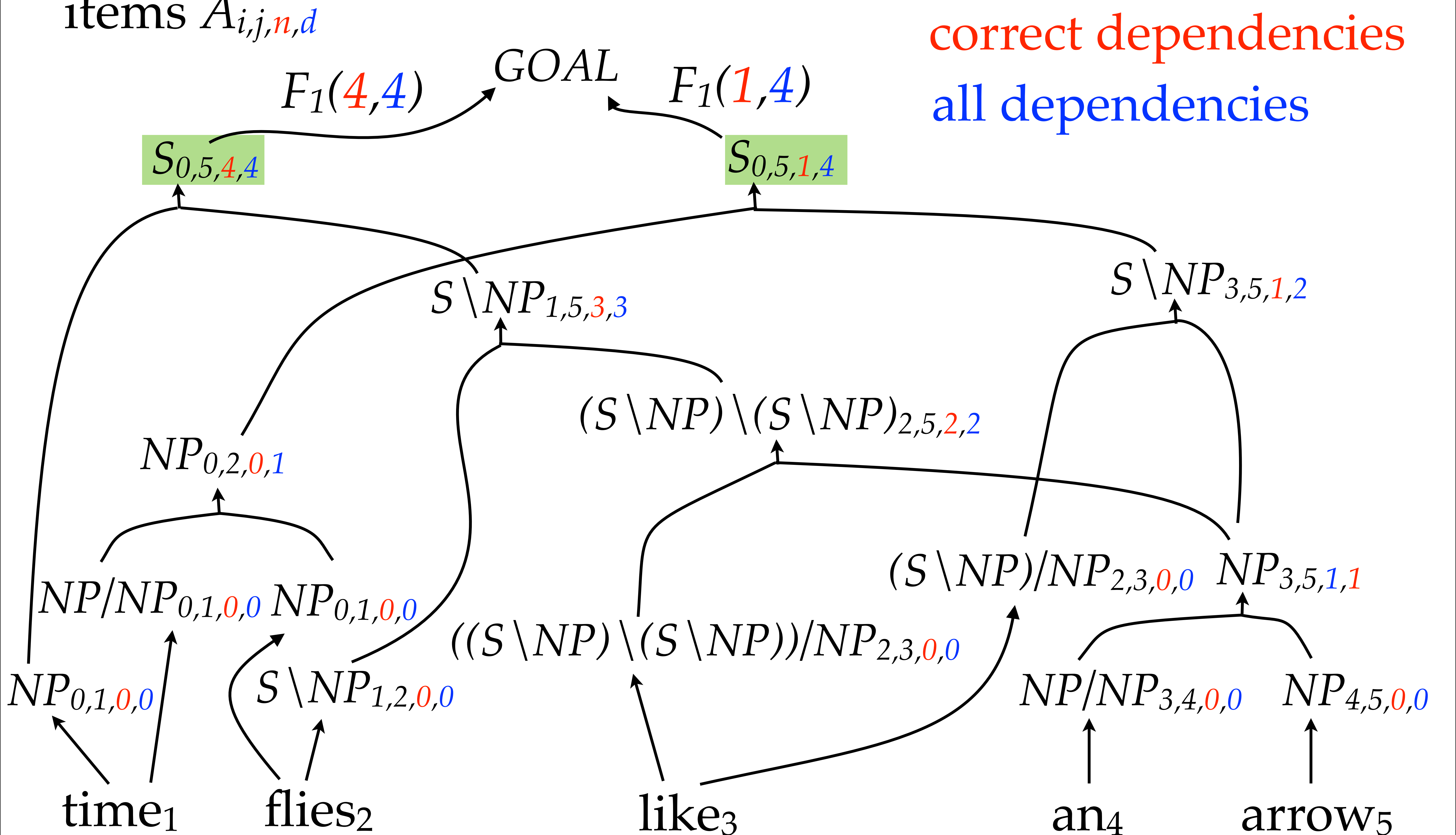
State-Split CKY

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



State-Split CKY

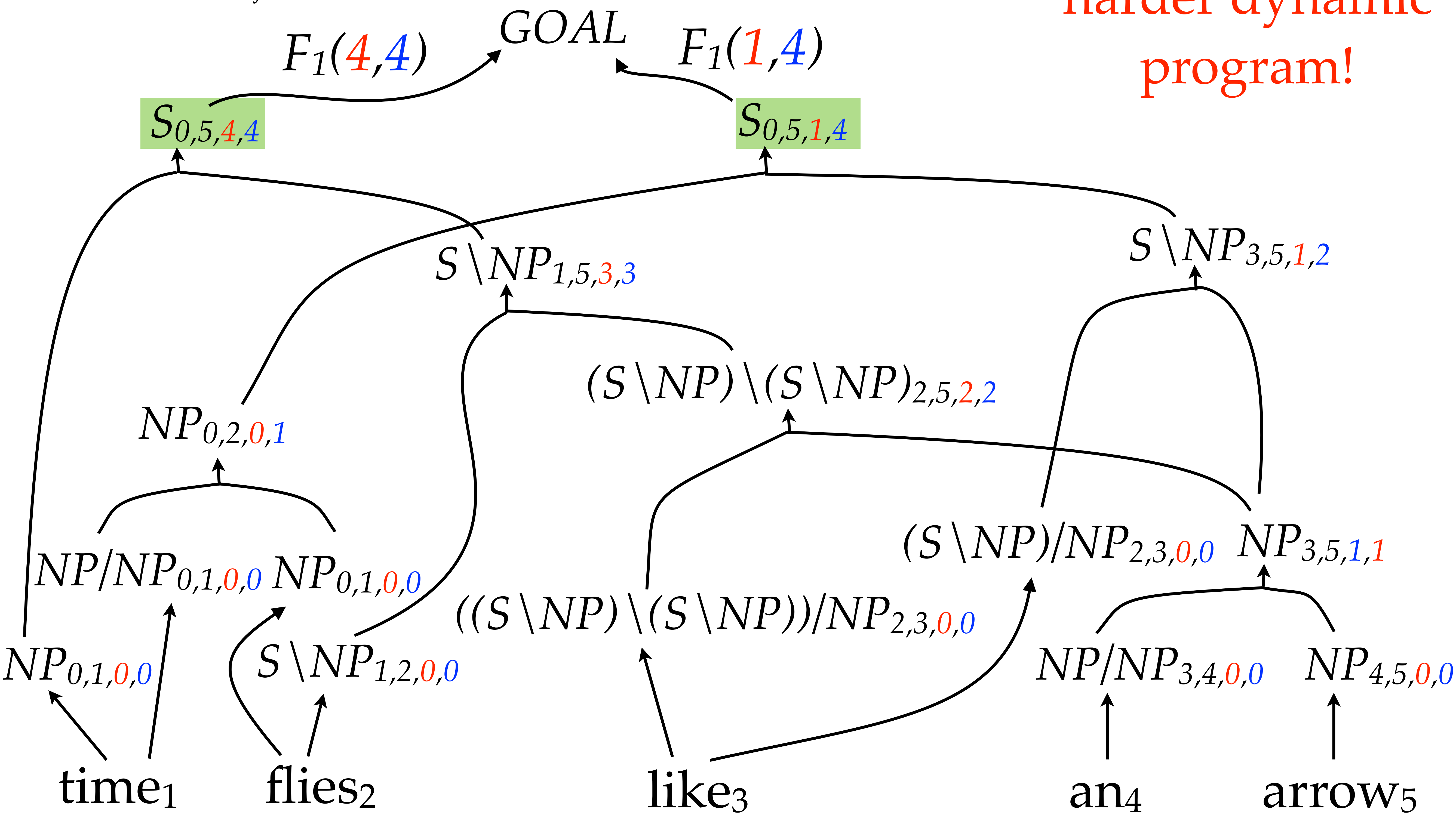
items $A_{i,j,n,d}$



State-Split CKY

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

harder dynamic program!



Experiments

- Exact versus approximate loss functions on test:

Loss	Approx	Exact

Experiments

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23

Experiments

- Exact versus approximate loss functions on test:

Loss	Approx	Exact
Precision	87.34	87.23
Recall	87.42	87.51

Experiments

- 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

- 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

- Results in large scale training setting on test:
- CLL
  - tight beam: 87.73
  - loose beam: 87.65

Experiments

- Results in large scale training setting on test:
- CLL
  - tight beam: 87.73
  - loose beam: 87.65
- DecF1
  - tight beam: 88.09
  - loose beam: **88.58**

Experiments

- Results in large scale training setting 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

Experiments

- Results with integrated model using BP:

CLL	87.65

Experiments

- Results with integrated model using BP:

CLL	87.65
BP	88.78

Experiments

- Results with integrated model using BP:

CLL	87.65
BP	88.78
+ $DecF_1$	89.06

Experiments

- Results with integrated model using BP:

CLL	87.65	
BP	88.78	
+ DecF$_1$	89.06	parser loss
+ SA	**89.24**	supertagger loss

Experiments

- Results with integrated model using BP:

CLL	87.65	
BP	88.78	
+ $DecF_1$	89.06	parser loss
+ SA	**89.24**	supertagger loss

Gains are additive!

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	86.73	

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74	
Petrov I-5	86.01	Fowler & Penn (2010)
BP	86.73	
+ $DecF_1$	87.02	

Experiments

- Auto-POS with integrated model using BP:

CLL	85.74	
Petrov I-5	86.01	Fowler & Penn (2010)
BP	86.73	
+ $DecF_1$	87.02	parser loss
+ SA	**87.17**	supertagger loss

Experiments

- What is the performance when speed is important?

Experiments

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

Experiments

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

CLL	87.73	
BP	88.20	
+ DecF_1	88.28	parser loss
+ SA	**88.46**	supertagger loss

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, gains are additive.

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, gains are additive.
- **Best reported results for CCG parsing.**

Future Directions

- Integration of POS sequence model.
- Grammar induction with combined model.
- Application to other grammar formalisms & problems.

Thanks!

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
David A. Smith
David Sontag
Mark Steedman
Charles Sutton

References

- A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. Michael Auli and Adam Lopez. To appear in *Proceedings of ACL,* June 2011.
- Efficient CCG Parsing: A* versus Adaptive Supertagging. Michael Auli and Adam Lopez. To appear in *Proceedings of ACL,* June 2011.