Learning Context-dependent Mappings from Sentences to Logical Form

Luke Zettlemoyer and Michael Collins

MIT Computer Science and Artificial Intelligence Lab

CSAIL

Context-dependent Analysis

```
Show me flights from New York to Singapore.

Which of those are nonstop?

Show me the cheapest one.

What about connecting?
```

Context-dependent Analysis

Show me flights from New York to Singapore.

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SIN)$

Which of those are nonstop?

Show me the cheapest one.

What about connecting?

Context-dependent Analysis

Show me flights from New York to Singapore.

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN)$

Which of those are nonstop?

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land nonstop(x)$

Show me the cheapest one.

What about connecting?

Context-dependent Analysis

Show me flights from New York to Singapore.

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SIN)$

Which of those are nonstop?

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SIN) \wedge nonstop(x)$

Show me the cheapest one.

$$argmax(\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SIN) \wedge nonstop(x), \\ \lambda y.cost(y))$$

What about connecting?

Context-dependent Analysis

`Show me flights from New York to Singapore.`

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN)$

`Which of those are nonstop?`

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land nonstop(x)$

`Show me the cheapest one.`

$argmax(\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land nonstop(x),$
$\quad \lambda y.cost(y))$

`What about connecting?`

$argmax(\lambda x.flight(x) \land from(x,NYC) \land to(x,SIN) \land connect(x),$
$\quad \lambda y.cost(y))$

A Supervised Learning Problem

Training Examples:
sequences of sentences and logical forms

```
Show me flights from New York to Seattle.
```
$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

```
List ones from Newark on Friday.
```
$\lambda x.flight(x) \wedge from(x,NEW) \wedge to(x,SEA) \wedge day(x,FRI)$

```
Show me the cheapest.
```
$argmax(\lambda x.flight(x) \wedge from(x,NEW) \wedge to(x,SEA) \wedge day(x,FRI),$
$\quad \lambda y.cost(y))$

A Supervised Learning Problem

Goal: Find a function f

$\lambda x.flight(x) \wedge from(x, NYC)$
$\wedge to(x, SEA)$

$\lambda x.flight(x) \wedge to(x, SEA)$
$\wedge from(x, NEW) \wedge day(x, FRI)$

`Show me the cheapest?`

f

$argmax(\lambda x.flight(x) \wedge from(x, NEW) \wedge to(x, SEA) \wedge day(x, FRI),$
$\lambda y.cost(y))$

A Supervised Learning Problem

Goal: Find a function f

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

`Show me the cheapest?`

f

$argmax(\lambda x.flight(x) \wedge from(x,NEW) \wedge to(x,SEA) \wedge day(x,FRI),$
$\lambda y.cost(y))$

Key Challenges:

- Structured input and output (lambda calculus)
- Hidden variables (only annotate final logical forms)

Talk Outline

- Sketch of the Approach
- Context-sensitive Derivations
- A Learning Algorithm
- Evaluation

An Example Analysis

Show me flights from New York to Seattle.

$\lambda x.\mathit{flight}(x) \wedge \mathit{from}(x, NYC) \wedge \mathit{to}(x, SEA)$

List ones from Newark on Friday.

An Example Analysis

Context:

$\lambda x.flight(x) \wedge from(x, NYC) \wedge to(x, SEA)$

Current sentence:

`List ones from Newark on Friday.`

An Example Analysis

Context:

$\lambda x.flight(x) \land from(x, NYC)$
$\land to(x, SEA)$

Current sentence:

`List ones from Newark on Friday.`

Step 1: Context-independent parse

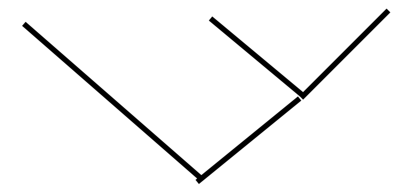
An Example Analysis

Context:

$\lambda x.flight(x) \wedge from(x,NYC)$
$\wedge to(x,SEA)$

Current sentence:

`List ones from Newark on Friday.`



$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

Step 1: Context-independent parse

An Example Analysis

Context:

$\lambda x.flight(x) \wedge from(x, NYC)$
$\wedge to(x, SEA)$

Current sentence:

`List ones from Newark on Friday.`

$\lambda x.\boxed{!f(x)} \wedge from(x, NEW) \wedge day(x, FRI)$

Step 1: Context-independent parse

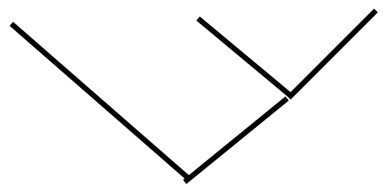
An Example Analysis

Context:

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

Current sentence:

`List ones from Newark on Friday.`



$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

Step 1: Context-independent parse

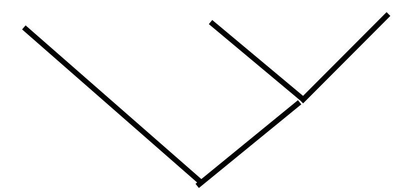
An Example Analysis

Context:

$\lambda x.flight(x) \wedge from(x, NYC)$
$\wedge to(x, SEA)$

Current sentence:

`List ones from Newark on Friday.`



$\lambda x.!f(x) \wedge from(x, NEW) \wedge day(x, FRI)$

Step 1: Context-independent parse

Step 2: Resolve reference

An Example Analysis

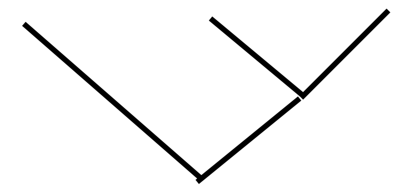
Context:

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA)$

Current sentence:

`List ones from Newark on Friday.`



$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

Step 1: Context-independent parse

Step 2: Resolve reference

An Example Analysis

Context:

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SEA)$

Current sentence:

`List ones from Newark on Friday.`

$\lambda x.!f(x) \land from(x,NEW) \land day(x,FRI)$

$\lambda x.flight(x) \land to(x,SEA)$

$\lambda x.flight(x) \land to(x,SEA) \land from(x,NEW) \land day(x,FRI)$

Step 1: Context-independent parse

Step 2: Resolve reference

Talk Outline

- Sketch of Approach
- Context-sensitive Derivations
- A Learning Algorithm
- Evaluation

Derivations

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Derivations

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Step 1: CCG Parsing

List	flights	to	Singapore
S/N	N	(N\N)/NP	NP
$\lambda f.f(x)$	$\lambda x.flight(x)$	$\lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y)$	sin

N\N

$\lambda f.\lambda x.f(x) \wedge to(x,sin)$

N

$\lambda x.flight(x) \wedge to(x,sin)$

S

$\lambda x.flight(x) \wedge to(x,sin)$

Step 1: CCG Parsing

List	flights	to	Singapore
S/N	N	(N\N)/NP	NP
$\lambda f.f(x)$	$\lambda x.flight(x)$	$\lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y)$	sin

N\N

$\lambda f.\lambda x.f(x) \wedge to(x,sin)$

N

$\lambda x.flight(x) \wedge to(x,sin)$

S

$\lambda x.flight(x) \wedge to(x,sin)$

Step 1: CCG Parsing

$$
\begin{array}{cccc}
\texttt{List} & \texttt{flights} & \texttt{to} & \texttt{Singapore} \\
\hline
S/N & N & (N\backslash N)/NP & NP \\
\lambda f.f(x) & \lambda x.flight(x) & \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y) & sin \\
\cline{3-4}
 & & \multicolumn{2}{c}{N\backslash N} \\
 & & \multicolumn{2}{c}{\lambda f.\lambda x.f(x) \wedge to(x,sin)} \\
\cline{2-4}
 & \multicolumn{3}{c}{N} \\
 & \multicolumn{3}{c}{\lambda x.flight(x) \wedge to(x,sin)} \\
\hline
\multicolumn{4}{c}{S} \\
\multicolumn{4}{c}{\lambda x.flight(x) \wedge to(x,sin)}
\end{array}
$$

Step 1: CCG Parsing

List	flights	to	Singapore
S/N	N	(N\N)/NP	NP
$\lambda f.f(x)$	$\lambda x.flight(x)$	$\lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y)$	sin

N\N

$\lambda f.\lambda x.f(x) \wedge to(x,sin)$

N

$\lambda x.flight(x) \wedge to(x,sin)$

S

$\lambda x.flight(x) \wedge to(x,sin)$

Step 1: CCG Parsing

List	flights	to	Singapore
S/N	N	$(N\backslash N)/NP$	NP
$\lambda f.f(x)$	$\lambda x.flight(x)$	$\lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y)$	sin

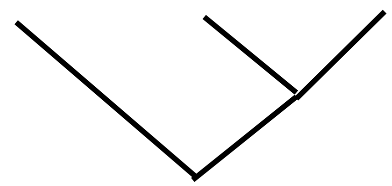
$$N\backslash N$$
$$\lambda f.\lambda x.f(x) \wedge to(x,sin)$$

$$N$$
$$\lambda x.flight(x) \wedge to(x,sin)$$

$$S$$
$$\lambda x.flight(x) \wedge to(x,sin)$$

Step 1: Referential lexical items

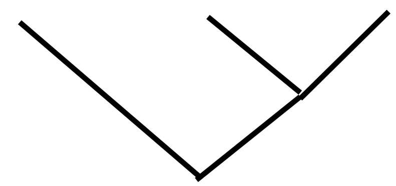
`List ones from Newark on Friday.`



$$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$$

Step 1: Referential lexical items

`List ones from Newark on Friday.`



$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

First extension:

- Add referential lexical items

ones	N	$\lambda x.!f(x)$
it	NP	$!e$

…

Step 1: Type-shifting operations

Second extension:

- Add type-shifting operators for elliptical expressions

```
the cheapest
```

Step 1: Type-shifting operations

Second extension:

- Add type-shifting operators for elliptical expressions

$$\frac{\texttt{the cheapest}}{\begin{array}{c}\mathbf{NP/N}\\ \lambda g.\mathit{argmin}(g, \lambda y.\mathit{cost}(y))\end{array}}$$

Step 1: Type-shifting operations

Second extension:

- Add type-shifting operators for elliptical expressions

$$\dfrac{\dfrac{\texttt{the cheapest}}{\begin{array}{c}\mathbf{NP/N} \\ \lambda g.\mathit{argmin}(g, \lambda y.\mathit{cost}(y))\end{array}}}{\begin{array}{c}\mathbf{NP} \\ \mathit{argmin}(\lambda x.!f(x), \lambda y.\mathit{cost}(y))\end{array}}$$

Step 1: Type-shifting operations

Second extension:

- Add type-shifting operators for elliptical expressions

$$\dfrac{\dfrac{\texttt{the cheapest}}{\begin{array}{c}\mathbf{NP/N}\\ \lambda g.\,argmin(g, \lambda y.\,cost(y))\end{array}}}{\begin{array}{c}\mathbf{NP}\\ argmin(\lambda x.\,!f(x), \lambda y.\,cost(y))\end{array}}$$

$$\texttt{A/B} : \texttt{g} \quad \texttt{=>} \quad \texttt{A} : \texttt{g}(\lambda x.\,!f(x))$$

where `g` is a function with input type `<e,t>`

Derivations

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \land from(x,NEW) \land day(x,FRI)$

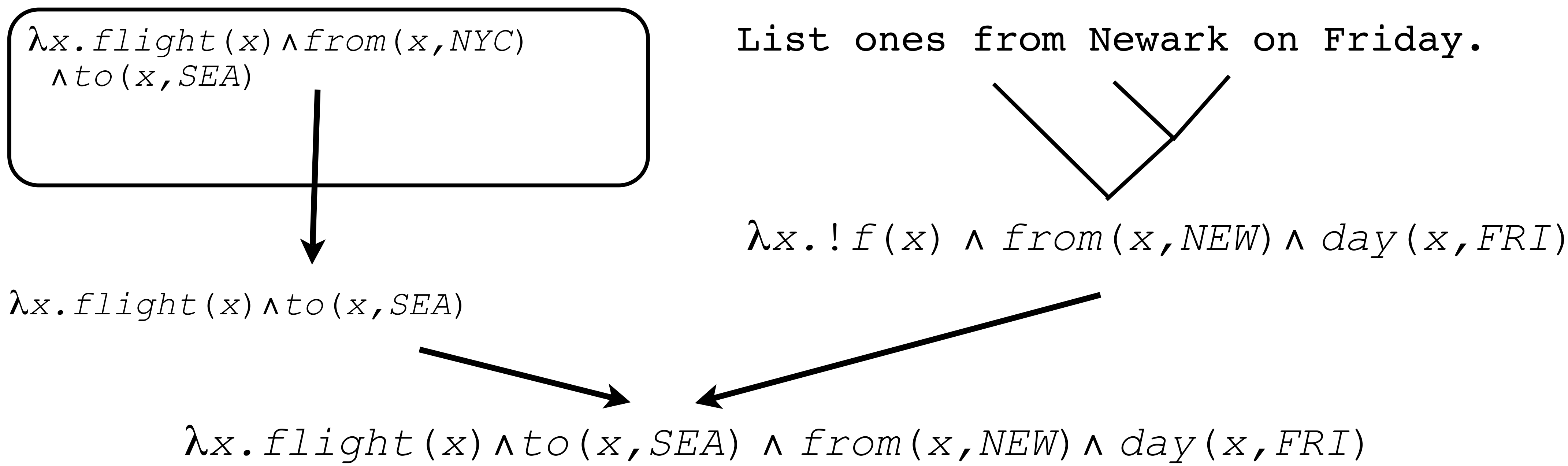
$\lambda x.flight(x) \land to(x,SEA)$

$\lambda x.flight(x) \land to(x,SEA) \land from(x,NEW) \land day(x,FRI)$

Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Derivations



Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Step 2: Resolving References

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \land from(x,NEW) \land day(x,FRI)$

$\lambda x.flight(x) \land to(x,SEA)$

$\lambda x.flight(x) \land to(x,SEA) \land from(x,NEW) \land day(x,FRI)$

For each reference:

- Select an expression from the context
- Substitute into current analysis

Step 2: Selecting from Context

For each logical form in context,
enumerate $\texttt{e}$ and $\texttt{<e,t>}$ type subexpressions:

Context:

$\lambda x.\mathit{flight}(x) \wedge \mathit{from}(x, \mathit{NYC}) \wedge \mathit{to}(x, \mathit{SEA})$

$\lambda x.\mathit{flight}(x) \wedge \mathit{to}(x, \mathit{SEA}) \wedge \mathit{from}(x, \mathit{NEW}) \wedge \mathit{day}(x, \mathit{FRI})$

$\mathit{argmax}(\lambda x.\mathit{flight}(x) \wedge \mathit{to}(x, \mathit{SEA}) \wedge \mathit{from}(x, \mathit{BOS}), \lambda y.\mathit{depart}(y))$

Step 2: Selecting from Context

For each logical form in context,
enumerate `e` and `<e,t>` type subexpressions:

Context:

$\lambda x.flight(x) \wedge from(x, NYC) \wedge to(x, SEA)$ → SEA

$\lambda x.flight(x) \wedge to(x, SEA) \wedge from(x, NEW) \wedge day(x, FRI)$

$argmax(\lambda x.flight(x) \wedge to(x, SEA) \wedge from(x, BOS), \lambda y.depart(y))$

Step 2: Selecting from Context

For each logical form in context,
enumerate e and <e,t> type subexpressions:

Context:

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

$argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,BOS), \lambda y.depart(y))$

SEA

NYC

Step 2: Selecting from Context

For each logical form in context,
enumerate `e` and `<e,t>` type subexpressions:

Context:

$\lambda x.flight(x) \wedge from(x, NYC)$
$\wedge to(x, SEA)$

$\lambda x.flight(x) \wedge to(x, SEA)$
$\wedge from(x, NEW) \wedge day(x, FRI)$

$argmax(\lambda x.flight(x) \wedge to(x, SEA)$
$\wedge from(x, BOS),$
$\lambda y.depart(y))$

SEA

NYC

$\lambda x.flight(x) \wedge from(x, NYC) \wedge to(x, SEA)$

Step 2: Selecting from Context

For each logical form in context,
enumerate `e` and `<e,t>` type subexpressions:

Context:

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

$argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,BOS), \lambda y.depart(y))$

SEA

NYC

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.from(x,NYC) \wedge to(x,SEA)$ $\quad$ $\lambda x.flight(x) \wedge to(x,SEA)$ $\quad$ $\lambda x.flight(x) \wedge from(x,NYC)$

Step 2: Selecting from Context

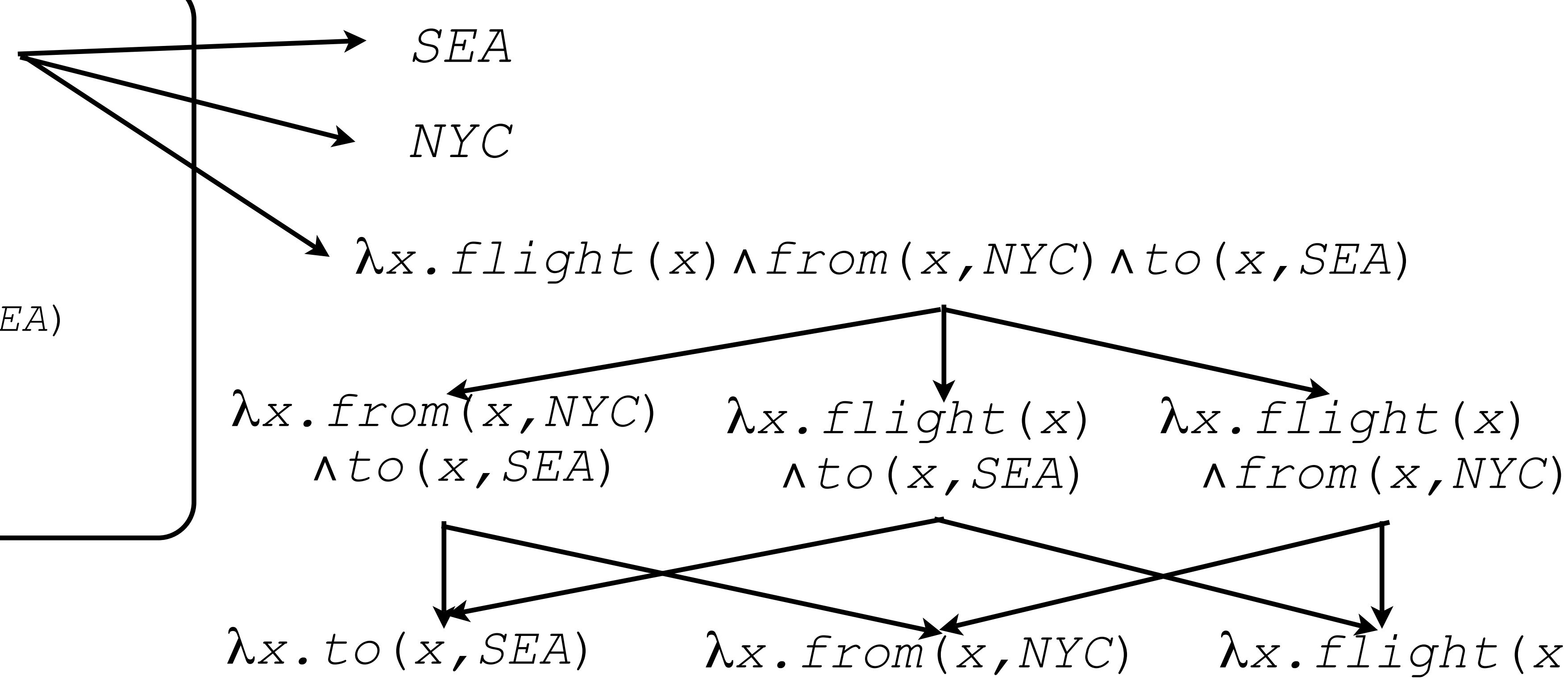
For each logical form in context,
enumerate `e` and `<e,t>` type subexpressions:

Context:

λx.flight(x) ∧ from(x,NYC) ∧ to(x,SEA)

λx.flight(x) ∧ to(x,SEA) ∧ from(x,NEW) ∧ day(x,FRI)

argmax(λx.flight(x) ∧ to(x,SEA) ∧ from(x,BOS), λy.depart(y))



Step 2: Resolving References

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

`List ones from Newark on Friday.`

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

For each reference:

- Select an expression from the context
- Substitute into current analysis

Derivations

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Derivations

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Three step process:

- Step 1: Context-independent parsing
- Step 2: Resolve all references
- Step 3: Optionally, perform an elaboration

Step 3: Elaboration operations

Show me the latest flight from New York to Seattle.

$argmax(\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA),$
$\quad \lambda y.time(y))$

on Friday

$argmax(\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA) \wedge day(x,FRI),$
$\quad \lambda y.time(y))$

Step 3: Elaboration operations

$argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge$
$from(x,NYC),$
$\lambda y.time(y))$

on Friday

$\downarrow$

$\lambda x.day(x,FRI)$

Step 3: Elaboration operations

$$argmax(\lambda x.flight(x) \land to(x,SEA) \land from(x,NYC), \lambda y.time(y))$$

`on Friday`

$$\lambda x.day(x,FRI)$$

$$\lambda f.argmax(\lambda x.flight(x) \land to(x,SEA) \land from(x,NYC) \land f(x), \lambda y.time(y))$$

Step 3: Elaboration operations

$$argmax(\lambda x.flight(x) \wedge to(x, SEA) \wedge from(x, NYC), \lambda y.time(y))$$

$$\downarrow$$

$$\boxed{\lambda f.}\, argmax(\lambda x.flight(x) \wedge to(x, SEA) \wedge from(x, NYC) \wedge \boxed{f(x),}\ \lambda y.time(y))$$

on Friday

$$\downarrow$$

$$\lambda x.day(x, FRI)$$

Step 3: Elaboration operations

$$argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NYC), \lambda y.time(y))$$

on Friday

$$\lambda x.day(x,FRI)$$

$$\boxed{\lambda f.}argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NYC) \wedge \boxed{f(x),} \lambda y.time(y))$$

$$argmax(\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA) \wedge day(x,FRI), \lambda y.time(y))$$

Step 3: Elaboration operations

$argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NYC), \lambda y.time(y))$

on Friday

$\lambda x.day(x,FRI)$

$\boxed{\lambda f.} argmax(\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NYC) \wedge \boxed{f(x),} \lambda y.time(y))$

$argmax(\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA) \wedge day(x,FRI), \lambda y.time(y))$

Possible elaborations:

- Potentially expand any embedded variable
- Can do deletions on elaboration function

Derivations

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

`List ones from Newark on Friday.`

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Three step process:

- **Step 1:** Context-independent parsing
- **Step 2:** Resolve all references
- **Step 3:** Optionally, perform an elaboration

Talk Outline

- Sketch of Approach
- Context-sensitive Derivations
- A Learning Algorithm
- Evaluation

Scoring Derivations

d

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

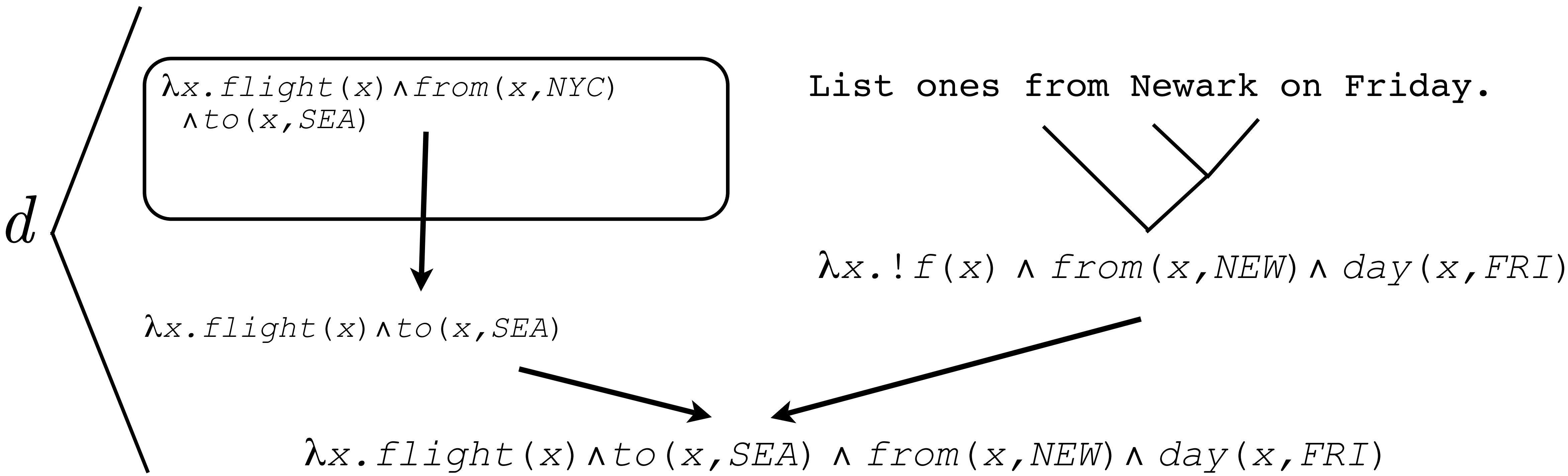
$\lambda x.flight(x) \wedge to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \wedge from(x,NEW) \wedge day(x,FRI)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

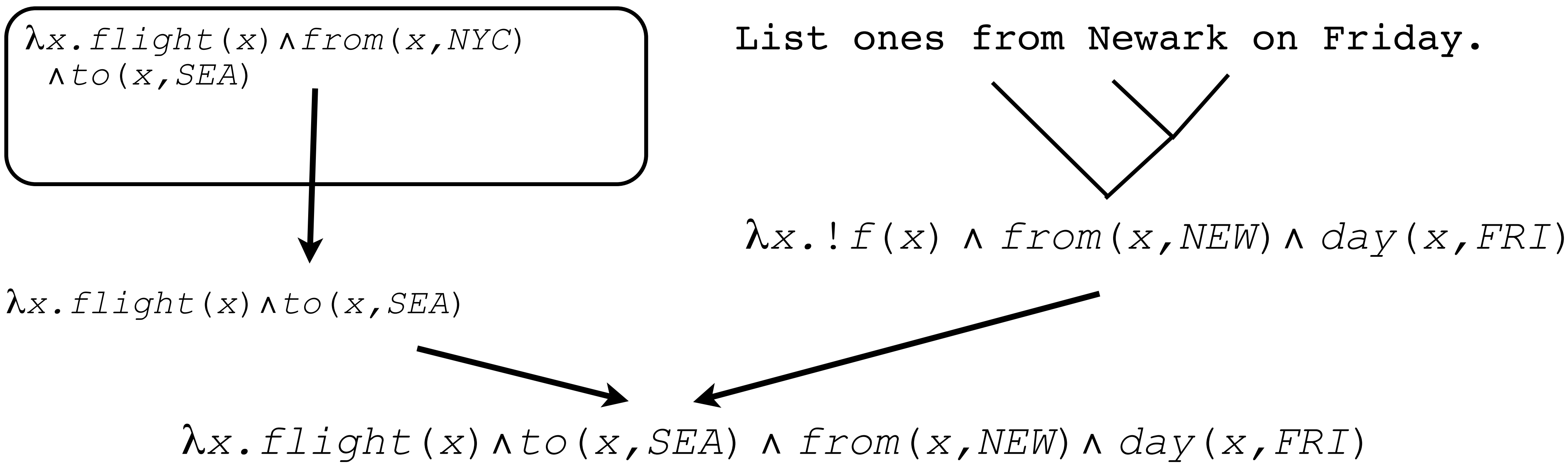
Scoring Derivations



Weighted linear model:

- Introduce features: $f(d)$
- Compute scores for derivations: $w \cdot f(d)$

Features for Derivations: $f(d)$



Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {`flight`, `from`, `to`, ...}
- Deletion indicators, for all pairs of predicates

  {(`from`, `flight`), (`from`, `from`), (`from`, `to`), ...}

Features for Derivations: $f(d)$

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \land from(x,NEW) \land day(x,FRI)$

$\lambda x.flight(x) \land to(x,SEA)$

$\lambda x.flight(x) \land to(x,SEA) \land from(x,NEW) \land day(x,FRI)$

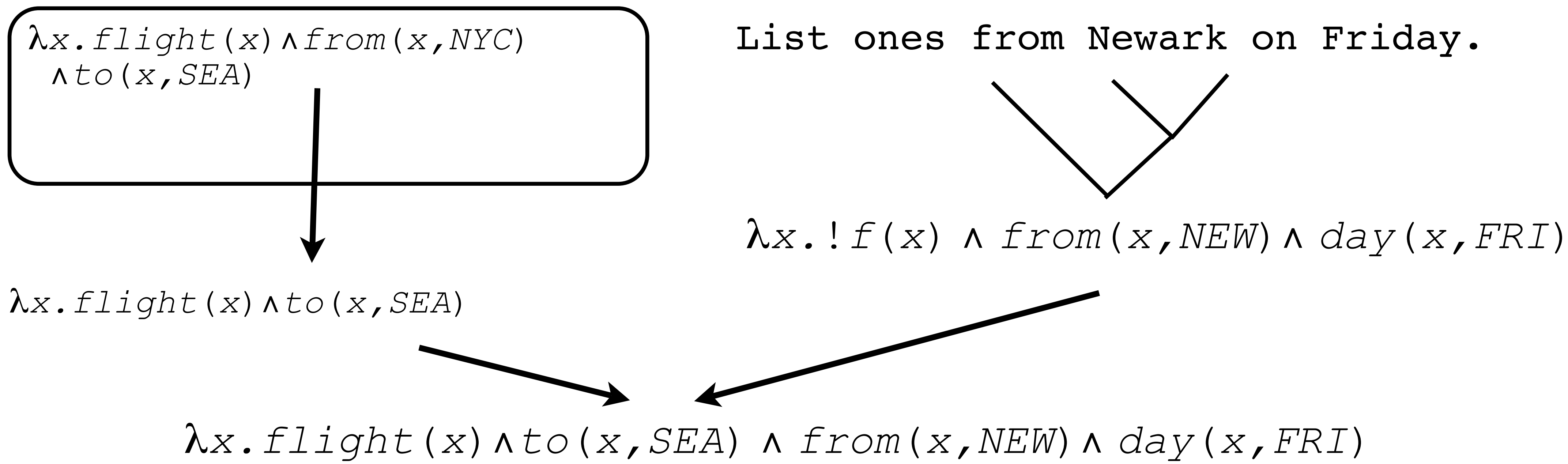
Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0, $\boxed{1}$, 2,...)
- Copy indicators, for all predicates {*flight*, *from*, *to*, ...}
- Deletion indicators, for all pairs of predicates

  {(*from*, *flight*), (*from*, *from*), (*from*, *to*), ...}

Features for Derivations: $f(d)$



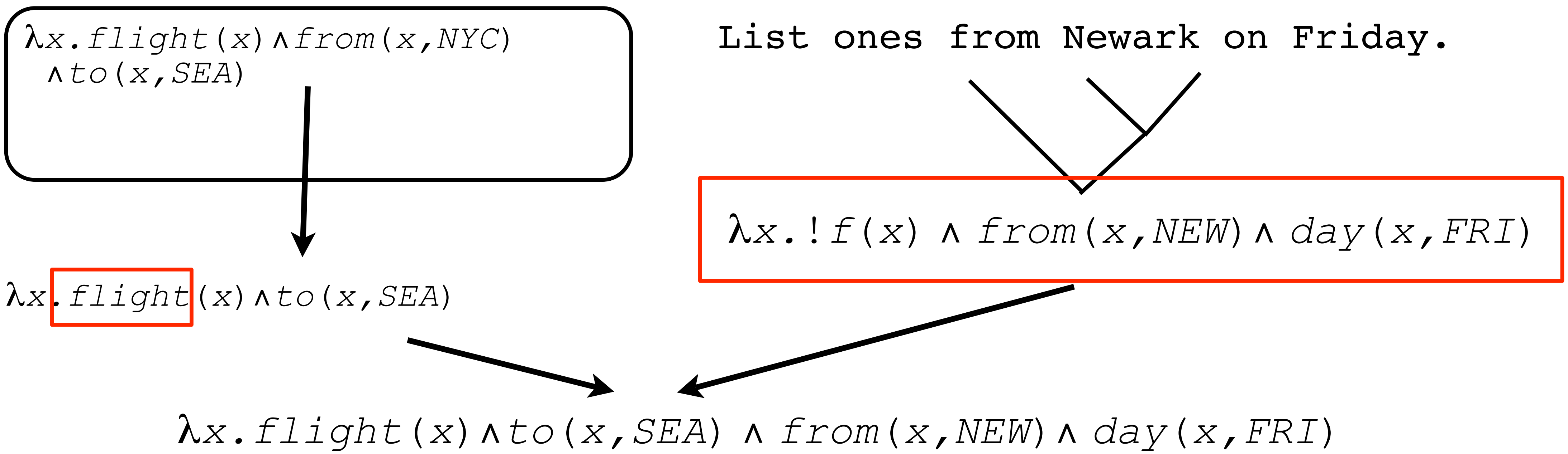
Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {*flight*, *from*, *to*, ...}
- Deletion indicators, for all pairs of predicates

  {(*from*, *flight*), (*from*, *from*), (*from*, *to*), ...}

Features for Derivations: $f(d)$



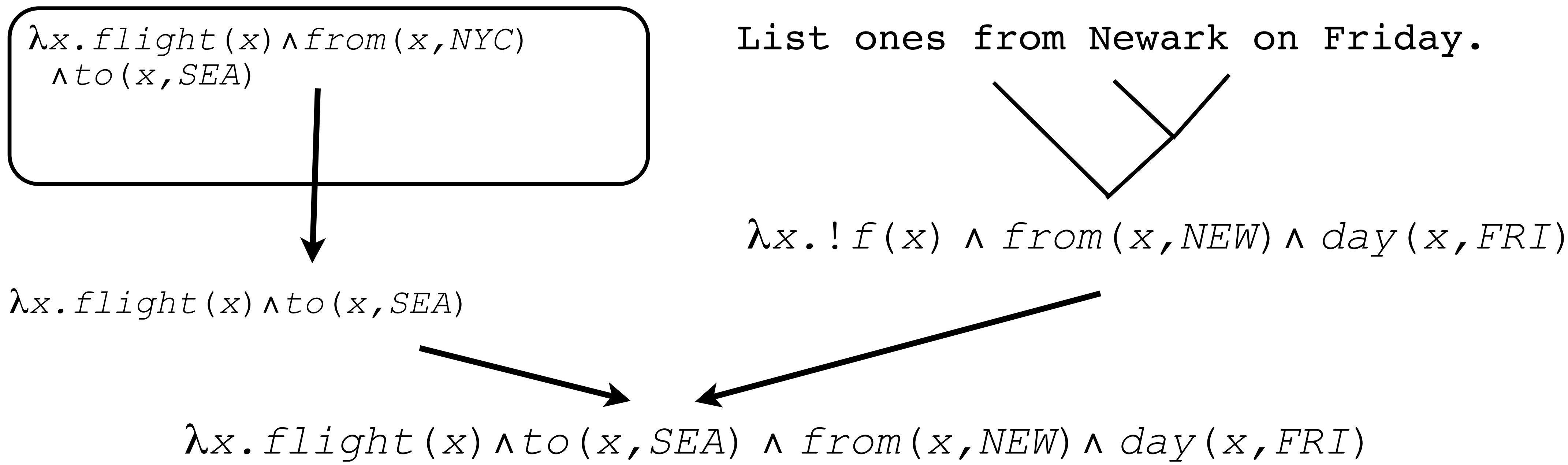
Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {$\boxed{\texttt{flight}}$, `from`, `to`, ...}
- Deletion indicators, for all pairs of predicates

  {(`from`, `flight`), (`from`, `from`), (`from`, `to`), ...}

Features for Derivations: $f(d)$



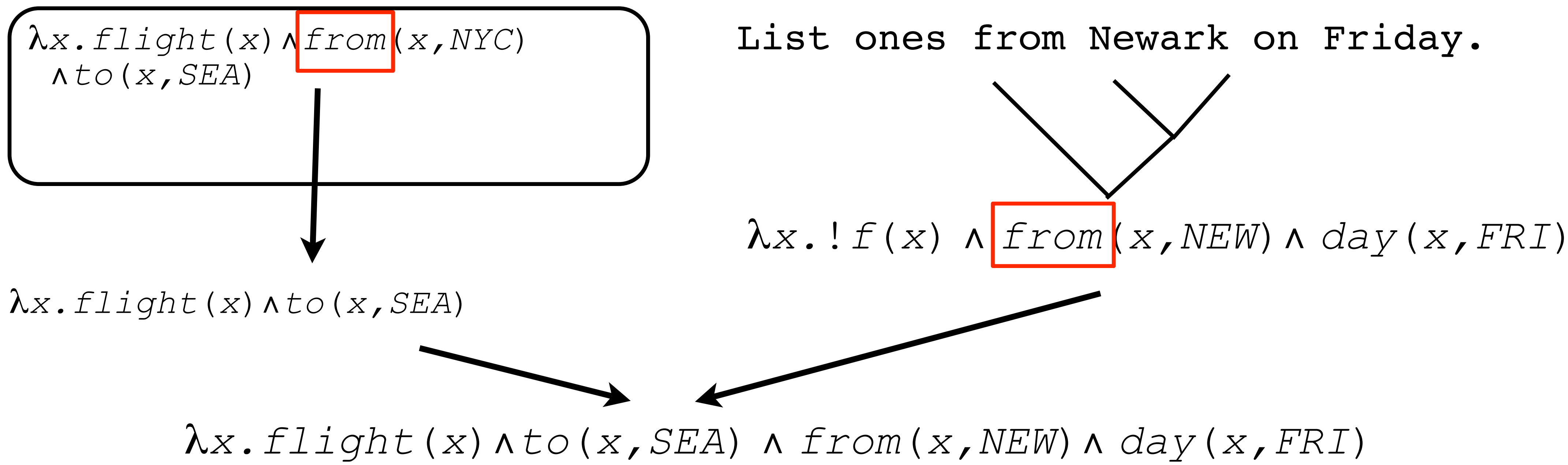
Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {`flight`, `from`, `to`, ...}
- Deletion indicators, for all pairs of predicates

  {(`from`, `flight`), (`from`, `from`), (`from`, `to`), ...}

Features for Derivations: $f(d)$



Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {`flight`, `from`, `to`, ...}
- Deletion indicators, for all pairs of predicates

  {(`from`, `flight`), (`from`, `from`), (`from`, `to`), ...}

Features for Derivations: $f(d)$

$\lambda x.flight(x) \land from(x,NYC) \land to(x,SEA)$

List ones from Newark on Friday.

$\lambda x.!f(x) \land from(x,NEW) \land day(x,FRI)$

$\lambda x.flight(x) \land to(x,SEA)$

$\lambda x.flight(x) \land to(x,SEA) \land from(x,NEW) \land day(x,FRI)$

Parsing features: set from Zettlemoyer and Collins (2007)

Context features:

- Distance indicators, for integers (0,1,2,...)
- Copy indicators, for all predicates {*flight*, *from*, *to*, ...}
- Deletion indicators, for all pairs of predicates

  {(*from*, *flight*), (*from*, *from*), (*from*, *to*), ...}

Inference and Learning

Two computations:

- Best derivation:

$$d^* = \arg\max_{d} \ w \cdot f(d)$$

- Best derivation with final logical form z :

$$d' = \arg \max_{d \ s.t. \ L(d)=z} w \cdot f(d)$$

We use a beam search algorithm.

Inference and Learning

Two computations:

- Best derivation:

$$d^* = \arg\max_d \; w \cdot f(d)$$

- Best derivation with final logical form z :

$$d' = \arg\max_{d \;\; s.t. \;\; L(d)=z} w \cdot f(d)$$

We use a beam search algorithm.

Learning:

- Hidden variable version of the structured perceptron algorithm [Liang et al., 2006] [Zettlemoyer & Collins, 2007]

Inputs: Training set $\{I_i \mid i = 1 \dots n\}$ of interactions. Each interaction $I = \{(w_{i,j}, z_{i,j}) \mid j = 1 \dots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1\ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j}, z_{i,j}) \mid j=1\ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1\ldots T,\ i=1\ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1\ldots n_i$: (Iterate training examples)

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1 \ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j}, z_{i,j}) \mid j=1 \ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1 \ldots T$, $i=1 \ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1 \ldots n_i$: (Iterate training examples)

Step 1: Check Correctness

- Find best analysis: $d^* = \arg\max_d w \cdot f(d)$
- If correct: $L(d^*) == z_{i,j}$, go to the Step 3.

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1\ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j},z_{i,j}) \mid j=1\ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1\ldots T,\ i=1\ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1\ldots n_i$: (Iterate training examples)

Step 1: Check Correctness

- Find best analysis: $d^* = \arg\max_d w \cdot f(d)$
- If correct: $L(d^*) == z_{i,j}$, go to the Step 3.

Step 3: Update context: Append $z_{i,j}$ to C

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1\ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j},z_{i,j}) \mid j=1\ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1\ldots T,\ i=1\ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1\ldots n_i$: (Iterate training examples)

Step 1: Check Correctness

- Find best analysis: $d^* = \arg\max_{d} w \cdot f(d)$
- If correct: $L(d^*) == z_{i,j}$, go to the Step 3.

Step 2: Update Parameters

- Find best correct analysis: $d' = \arg \max_{d \ s.t.\ L(d)=z_{i,j}} w \cdot f(d)$
- Update parameters: $w = w + f(d') - f(d^*)$

Step 3: Update context: Append $z_{i,j}$ to C

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1 \ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j}, z_{i,j}) \mid j=1 \ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1 \ldots T, i=1 \ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1 \ldots n_i$: (Iterate training examples)

Step 1: Check Correctness

- Find best analysis: $d^* = \arg\max_d w \cdot f(d)$
- If correct: $L(d^*) == z_{i,j}$, go to the Step 3.

Step 2: Update Parameters

- Find best correct analysis: $\boxed{d' = \arg \max_{d \ s.t. \ L(d)=z_{i,j}} w \cdot f(d)}$
- Update parameters: $w = w + f\boxed{(d')} - f(d^*)$

Step 3: Update context: Append $z_{i,j}$ to C

Output: Parameters w.

Inputs: Training set $\{I_i \mid i=1\ldots n\}$ of interactions. Each interaction $I=\{(w_{i,j},z_{i,j}) \mid j=1\ldots n_i\}$ is a sequence of sentences and logical forms. Initial parameters w. Number of iterations T.

Computation:

For $t=1\ldots T, i=1\ldots n$: (Iterate interactions)

Set $C=\{\}$ (Reset Context)

For $j=1\ldots n_i$: (Iterate training examples)

Step 1: Check Correctness

- Find best analysis: $\boxed{d^* = \arg\max_d \ w \cdot f(d)}$
- If correct: $L(d^*) == z_{i,j}$, go to the Step 3.

Step 2: Update Parameters

- Find best correct analysis: $\boxed{d' = \arg \max_{d \ s.t. \ L(d)=z_{i,j}} w \cdot f(d)}$
- Update parameters: $w = w + f\boxed{(d')} - f\boxed{(d^*)}$

Step 3: Update context: Append $z_{i,j}$ to C

Output: Parameters w.

Talk Outline

- Sketch of Approach
- Context-sensitive Derivations
- A Learning Algorithm
- Evaluation

Evaluation

- Domain: ATIS travel database queries
  - 399 training interactions (3813 sentences)
  - 127 test interactions (826 sentences)
- Comparison: previous state-of-the-art [Miller et al. 1996]
  - requires full annotation of all syntactic, semantic, and context-resolution decisions
  - decision tree learning

The Miller et al. [1996] Approach

Step 1: Semantic parsing

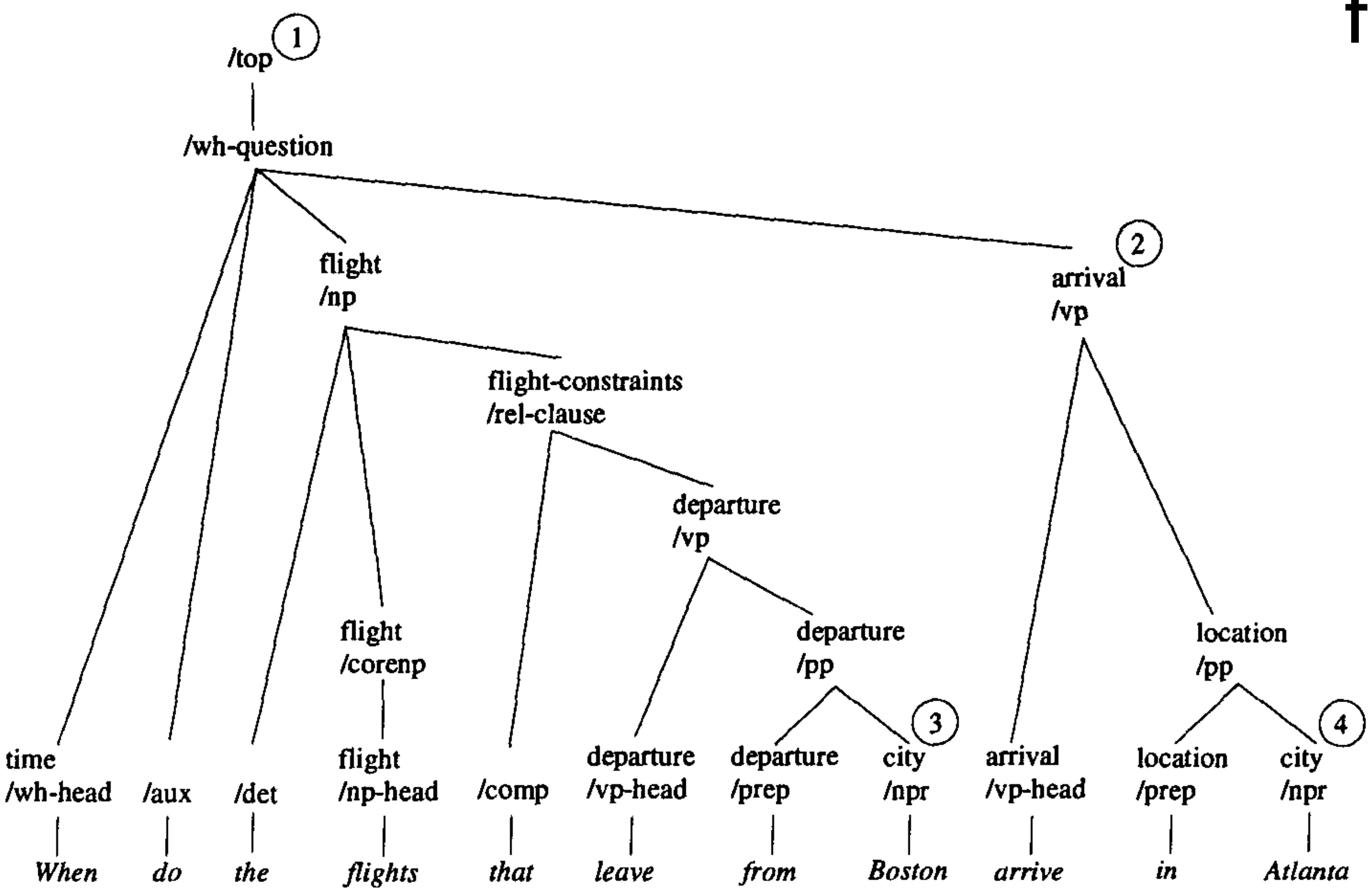


Figure 2: A sample parse tree.

Step 2: Select frame and fill slot values

Air-Transportation

Show: (Arrival-Time)

Origin: (City "Boston")

Destination: (City "Atlanta")

Figure 3: A sample semantic frame.

Step 3: Optionally copy slot values from previous frames

Evaluation

- Domain: ATIS travel database queries
  - 399 training interactions (3813 sentences)
  - 127 test interactions (826 sentences)
- Comparison: previous state-of-the-art [Miller et al. 1996]
- Metric: accuracy recovering fully correct meanings

Evaluation

- Domain: ATIS travel database queries
  - 399 training interactions (3813 sentences)
  - 127 test interactions (826 sentences)
- Comparison: previous state-of-the-art [Miller et al. 1996]
- Metric: accuracy recovering fully correct meanings
- Result: improved accuracy
  - 78.4% => 83.7%
  - less engineering effort: only annotated final meanings

Varying the Length of a Context Window M

ATIS Development Set:

Context Length	Accuracy
M=0	45.4
M=1	79.8
M=2	81.0
M=3	**82.1**
M=4	81.6
M=10	81.4

Example Learned Feature Weights

Negative weights:

- Distance features: (1,2,3,...)

Positive weights:

- Copy features: `flight, from, to`
- Deletion features: (`from, from`), (`nonstop, connect`), (`during-day, time`)

Summary

$\lambda x.flight(x) \wedge from(x,NYC) \wedge to(x,SEA)$

$\lambda x.flight(x) \wedge to(x,SEA) \wedge from(x,NEW) \wedge day(x,FRI)$

Show me the cheapest?

f

$$argmax(\lambda x.flight(x) \wedge from(x,NEW) \wedge to(x,SEA) \wedge day(x,FRI), \lambda y.cost(y))$$

Key challenges:

- Structured input and output, hidden structure not annotated

Solution:

- Analysis: two-stage approach
- Learn: how to incorporate meaning from the context

The End