Learning Expressive Ontological Concept Descriptions via Neural Networks

Marco Rospocher

FONDAZIONE BRUNO KESSLER

4th Workshop on Semantic Deep Learning (SemDeep-4)

This workshop will be held at ISWC 2018 (8-12 October in Monterey, California)

Previous editions: SemDeep-1@ESWC 2017, SemDeep-2@IWCS 2017, SemDeep-3@COLING 2018

First things first…

University of Trento - September 21, 2018

First things first…

Marco Rospocher
advisors
Chiara Ghidini

Giulio Petrucci
new Ph.D.

University of Trento - September 21, 2018

First things first…

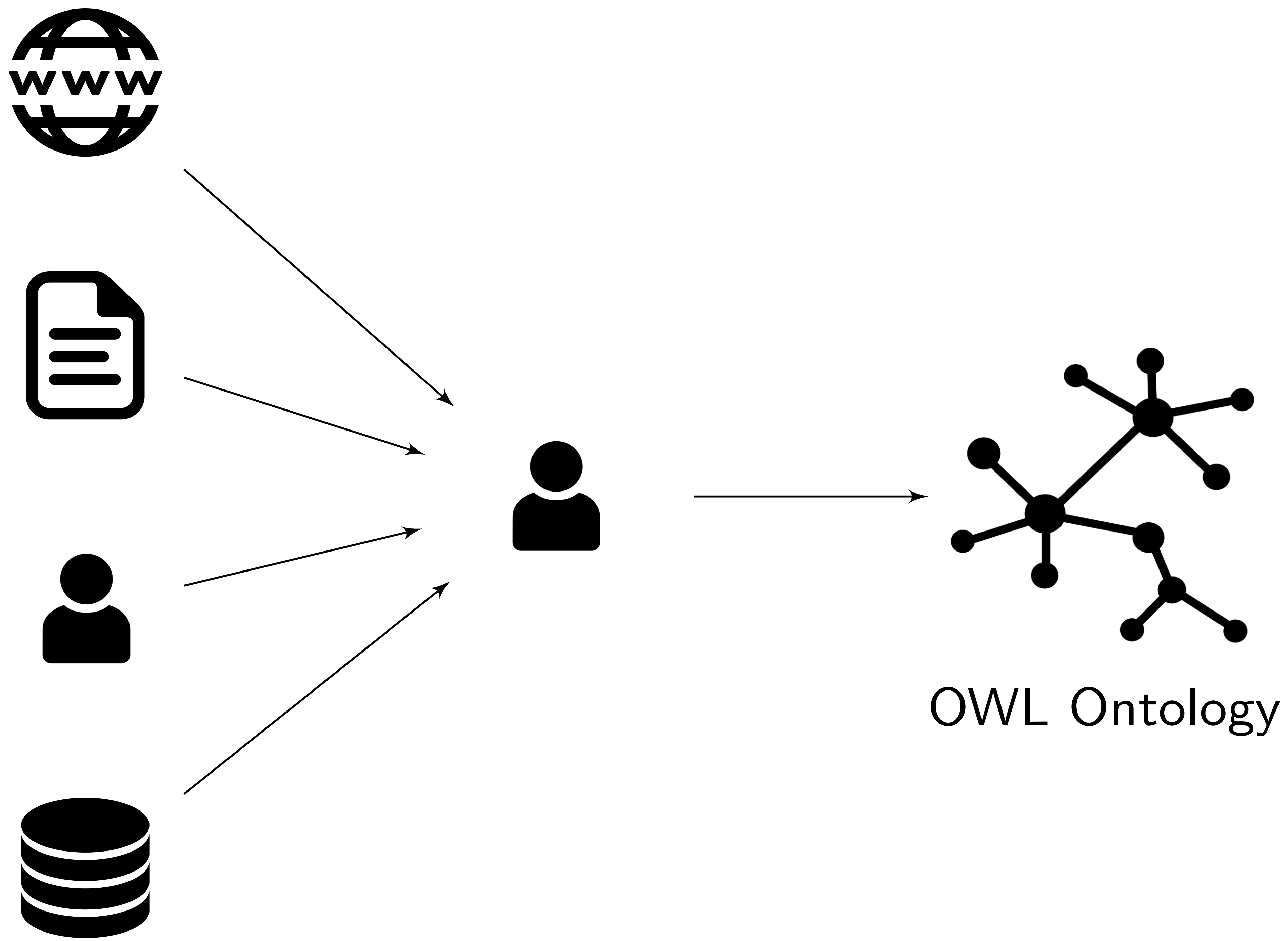
Most of following slides are taken from Giulio's defense

Marco Rospocher
advisors
Chiara Ghidini

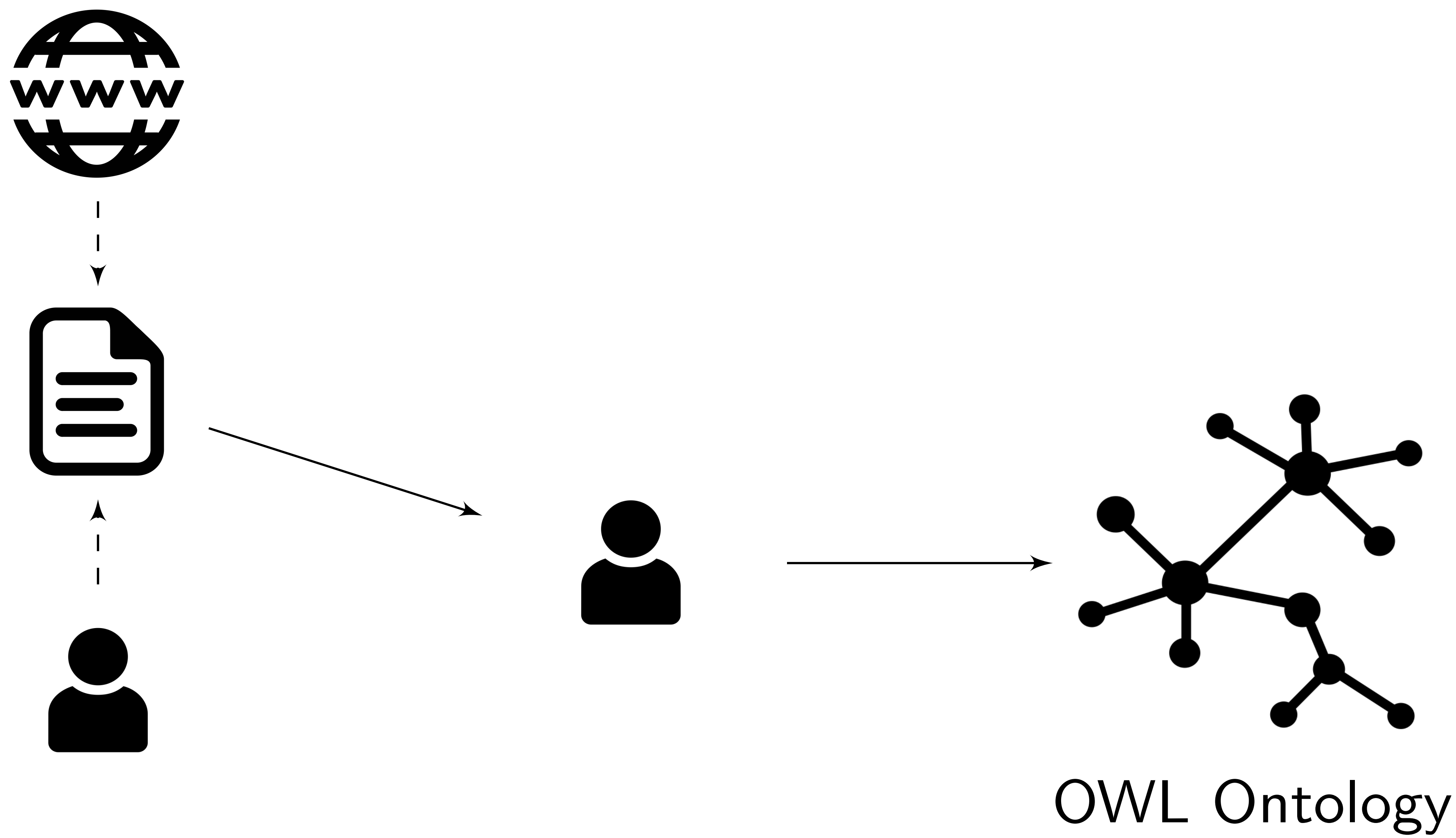
Giulio Petrucci
new Ph.D.

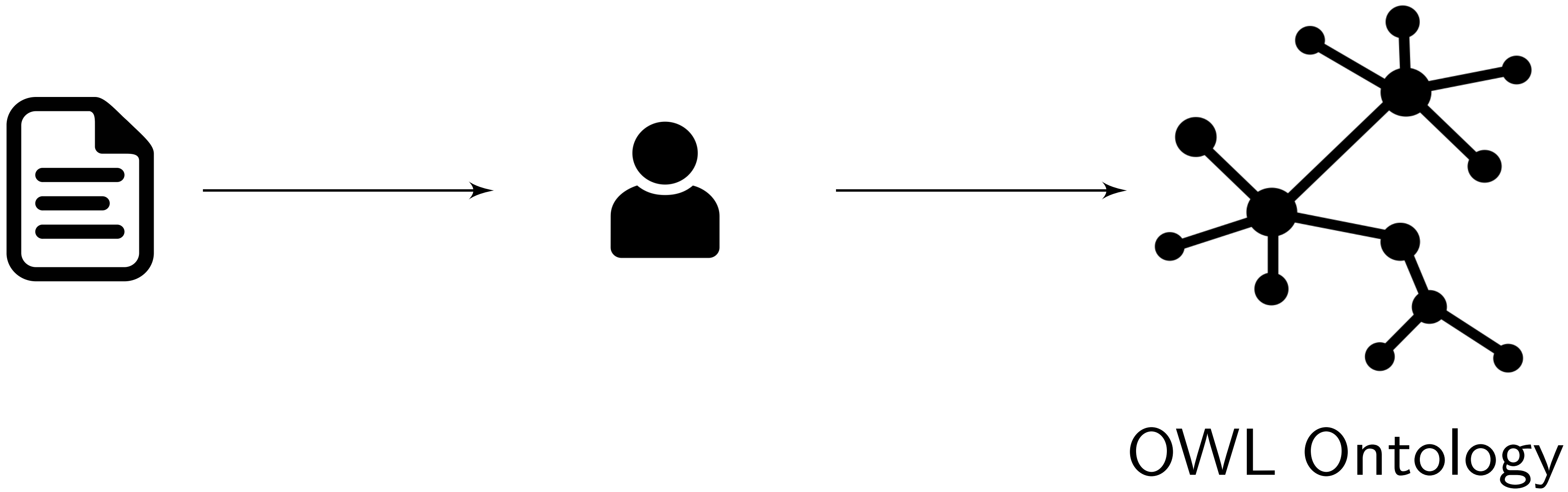
University of Trento - September 21, 2018

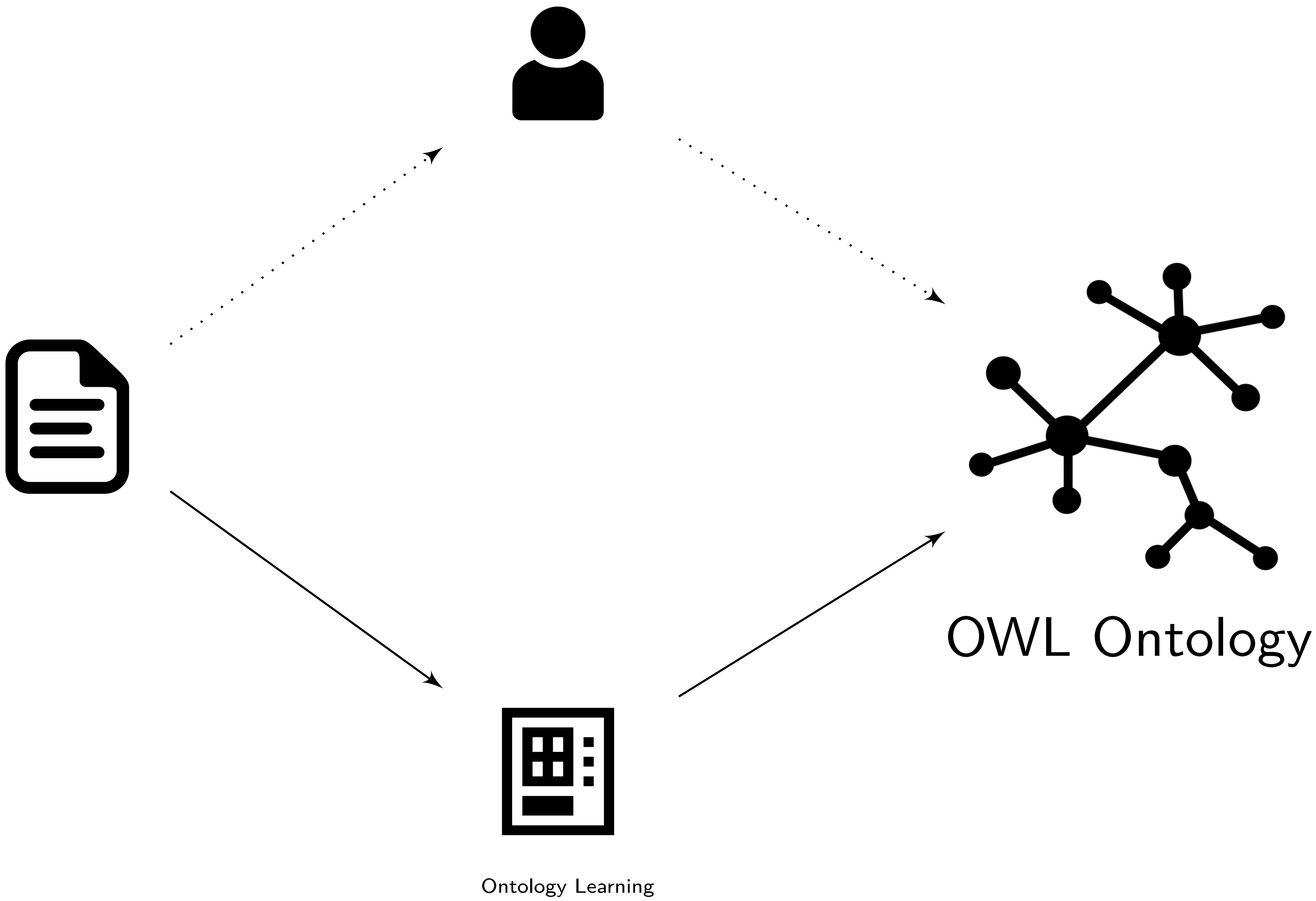
Ontology Engineering



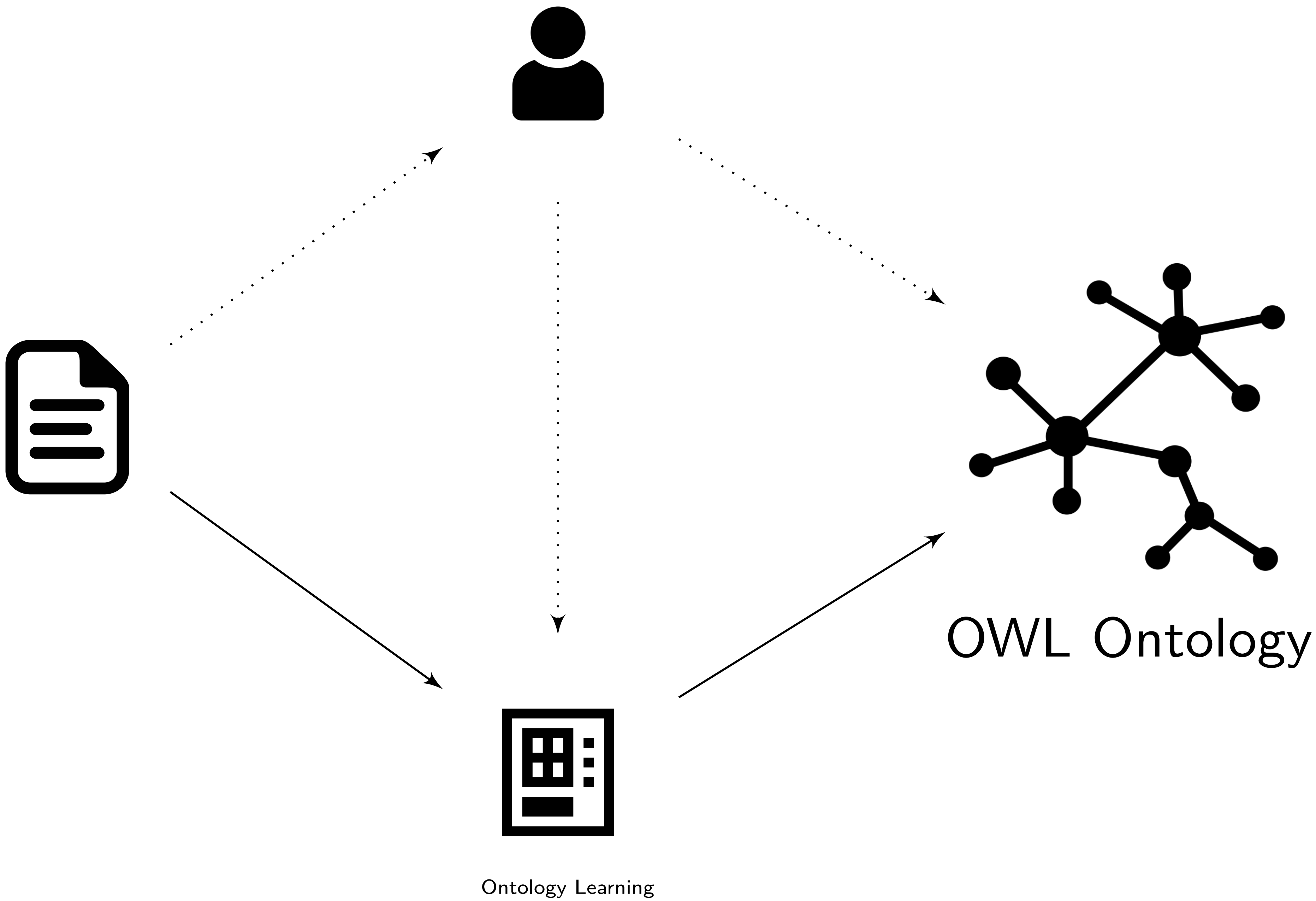
Ontology Engineering



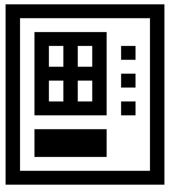

OWL Ontology


OWL Ontology
Ontology Learning

Ontology Engineering



Ontology Learning from Text

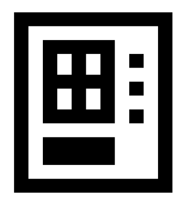


Ontology Learning from Text

Bees are insects that produce honey. They have six legs. Bees live only in beehives—or just hives. Maya and Flip are bees. Maya, in particular, is a notable bee. Maya and Flip are friends.

Ontology Learning from Text

Bees are insects that produce honey. They have six legs. Bees live only in beehives—or just hives. Maya and Flip are bees. Maya, in particular, is a notable bee. Maya and Flip are friends.



Ontology Learning

Terminological Knowledge about concepts, their definitions, and relations among them. Examples are:

- bees are insects;
- bees produce honey;
- bees have 6 legs;
- bees live in beehives.

Ontology Population

Asserional Knowledge about individuals, their mutual relations and their relations with concepts. Examples are:

- Maya is a bee;
- Flip is a bee;
- Maya and Flip are friends;

Ontology Learning from Text

Bees are insects that produce honey. They have six legs. Bees live only in beehives—or just hives. Maya and Flip are bees. Maya, in particular, is a notable bee. Maya and Flip are friends.

Ontology Learning

Terminological Knowledge about concepts, their definitions, and relations among them. Examples are:

- bees are insects;
- bees produce honey;
- bees have 6 legs;
- bees live in beehives.

Ontology Population

Asserional Knowledge about individuals, their mutual relations and their relations with concepts. Examples are:

- Maya is a bee;
- Flip is a bee;
- Maya and Flip are friends;

Ontology Learning from Text

Bees are insects that produce honey. They have six legs. Bees live only in beehives—or just hives. Maya and Flip are bees. Maya, in particular, is a notable bee. Maya and Flip are friends.

Axiom	$\mathtt{Bee} \sqsubseteq \mathtt{Insect} \sqcap \exists \mathtt{produce.Honey}$
Relation	`produce(Bee, Honey)`
Hierarchy	`is_a(Bee, Insect)`
Concept	`Beehive`
Synonym	`{beehive, hive}`
Term	`bee, beehive, hive, honey, ...`

Table: Ontology Learning Layer Cake [1]

[1] Cimiano et al., 2009.

State of the Art

Up to 2007:

> *[...] state-of-the-art in lexical Ontology learning is able to generate ontologies that are largely informal or lightweight ontologies in the sense that they are limited in their expressiveness.*
> *— Völker et al, 2007.*

From 2008:

- LExO (Völker et al, 2008);
- LearningDL (Ma et al., 2014);
- TEDEI (Mathews et al., 2017);
- Gyawali et al., 2017.

State of the Art

Some common traits:

- heavily hand-crafted rules;
- relying on pre-trained NLP toolkits output to represent text;
- targeting different source and target languages

State of the Art

Some common traits:

- heavily hand-crafted rules;
- relying on pre-trained NLP toolkits output to represent text;
- targeting different source and target languages

Some common limitations:

- rigidity;
- cost in maintenance and evolution.

The Road Less Traveled

Transforming a sentence into an axiom:

The Road Less Traveled

Transforming a sentence into an axiom:

- is it possible to train a machine learning model for this task?

Transforming a sentence into an axiom:

- is it possible to train a machine learning model for this task?
- is it possible to perform the training in a end-to-end fashion?

Framing the Problem

$$\text{Natural Language} \xrightarrow{\text{transformation function}} \text{OWL Axiom}$$

Framing the problem: the source language

Navigli et al., 2010 address the problem of defining a definition as:

- `DEFINIENDUM (DF)`: the concept being defined (e.g., *"a bee"*);
- `DEFINITOR (VF)`: that introduces the definition (e.g., *"is"*);
- `DEFINIENS (GF)`: the *genus* phrase (e.g., *"an insect"*);
- `REST (DF)`: the *differentia* with respect to the genus (e.g., *"that produces honey"*).

A bee is an insect that produces honey.

Framing the problem: the source language

Navigli et al., 2010 address the problem of defining a definition as:

- `DEFINIENDUM (DF)`: the concept being defined (e.g., *"a bee"*);
- `DEFINITOR (VF)`: that introduces the definition (e.g., *"is"*);
- `DEFINIENS (GF)`: the *genus* phrase (e.g., *"an insect"*);
- `REST (DF)`: the *differentia* with respect to the genus (e.g., *"that produces honey"*).

A bee is an insect.

Framing the problem: the source language

Navigli et al., 2010 address the problem of defining a definition as:

- `DEFINIENDUM (DF)`: the concept being defined (e.g., *"a bee"*);
- `DEFINITOR (VF)`: that introduces the definition (e.g., *"is"*);
- `DEFINIENS (GF)`: the *genus* phrase (e.g., *"an insect"*);
- `REST (DF)`: the *differentia* with respect to the genus (e.g., *"that produces honey"*).

A bee produces honey.

Framing the Problem: the source language

Descriptive Language

- *A bee is an insect that produces honey.*
- *A bee is an insect.*
- *A bee produces honey.*

Framing the problem: the target language

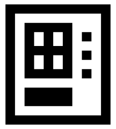
Description Logic languages provide primitives to represent application domains in terms of their relevant *concepts*, entities and *relations* among them.

Framing the problem: the target language

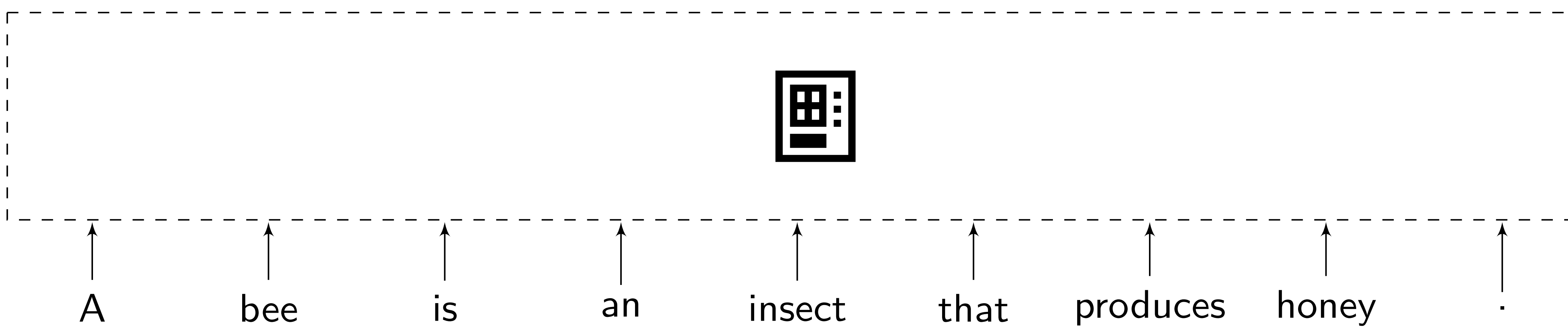
Description Logic languages provide primitives to represent application domains in terms of their relevant *concepts*, entities and *relations* among them. In particular, $\mathcal{ALCQ}$.

primitive	**syntax**	**semantics**
Universal concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset^{\mathcal{I}}$
Atomic concept	A	$A^{\mathcal{I}}$
Concept negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Concept intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Concept union ($\mathcal{U}$)	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Atomic role	R	$R^{\mathcal{I}}$
Value restriction	$\forall R.C$	$a \in \Delta^{\mathcal{I}} \mid \forall\, b\ .\ (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}$
Limited existential quantification	$\exists R.\top$	$a \in \Delta^{\mathcal{I}} \mid \exists\, b\ .\ (a, b) \in R^{\mathcal{I}}$
Full existential quantification ($\mathcal{E}$)	$\exists R.C$	$a \in \Delta^{\mathcal{I}} \vert \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}$
Unqualified numbered restriction ($\mathcal{N}$)	$\geqslant nR$	$a \in \Delta^{\mathcal{I}} \mid \vert\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}}\}\vert \geq n$
Qualified numbered restriction ($\mathcal{Q}$)	$\geqslant nR.C$	$a \in \Delta^{\mathcal{I}} \vert \vert\{b \in \Delta^{\mathcal{I}} \vert (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert \geq n$

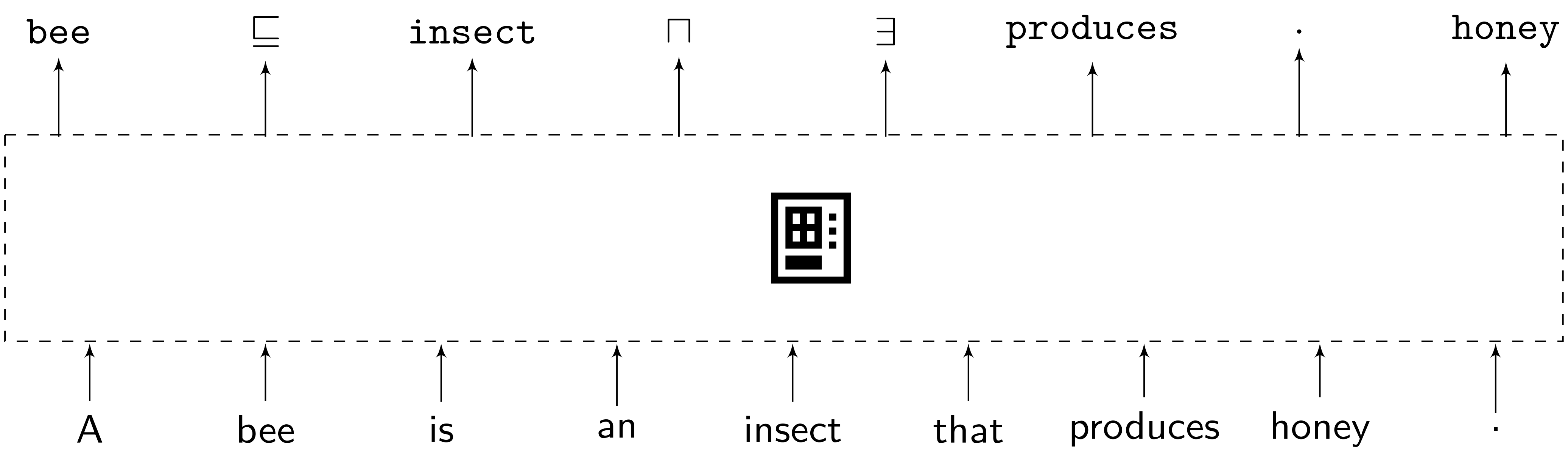
Framing the Problem: the Transformation Function



Framing the Problem: the Transformation Function

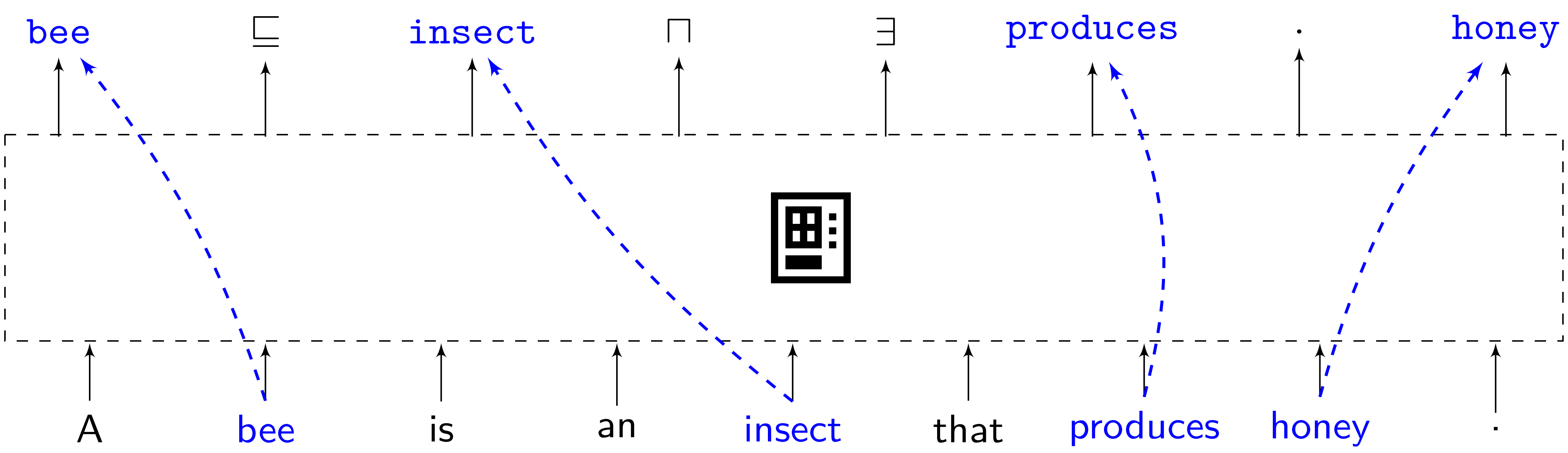


Framing the Problem: the Transformation Function



Framing the Problem: the Transformation Function

"syntactic transformation of natural language definitions into description logic axioms." (Völker J., 2008)



All the extralogical symbols come from the sentence.

Framing the Problem

$$\text{Descriptive Language} \xrightarrow{\text{syntactic transformation}} \mathcal{ALCQ}$$

We need:

- datasets;
- architecture;

Structure and Meaning

Machine Learning means examples, good examples, many examples.

Structure and Meaning

Machine Learning means examples, good examples, many examples.

- Every bee is an insect and it also produces honey.
- A bee is an insect that produces honey.
- Bees are insects that produce also honey.

Structure and Meaning

Machine Learning means examples, good examples, many examples.

- Every bee is an insect and it also produces honey.
- A bee is an insect that produces honey.
- Bees are insects that produce also honey.

- Every bee is an insect that produces also honey.

Structure and Meaning

Machine Learning means examples, good examples, many examples.

- Every bee is an insect and it also produces honey.
- A bee is an insect that produces
- Bees are insects that produce also honey.

- Every bee is an insect that produces also honey.

Structure and Meaning

Machine Learning means examples, good examples, many examples.

- Every bee is an insect and it also produces honey.
- A bee is an insect that produces honey.
- Bees are insects that produce also honey.

- Every bee is an insect that produces also honey.

Many structures, one meaning.

Meaning and Structure

Machine Learning means examples, good examples, many examples.

Meaning and Structure

Machine Learning means examples, good examples, many examples.

- Bees are insects that produce honey.
- A bee is also an insect that produces honey.
- Every bee is an insect and it also produces honey.
- A cow is a mammal that eats grass.

Meaning and Structure

Machine Learning means examples, good examples, many examples.

- Bees are insects that produce honey.
- A bee is also an insect that produces honey.
- Every bee is an insect and it also produces honey.
- A cow is a mammal that eats grass.

- A cow is a mammal that produces milk.

Meaning and Structure

Machine Learning means examples, good examples, many examples.

- Bees are insects that produce honey.
- A bee is also an insect that produces honey.
- Every bee is an insect and it also produces honey.
- A cow is a mammal that eats grass.

- A cow is a mammal that produces milk.

Machine Learning means examples, good examples, many examples.

- Bees are insects that produce honey.
- A bee is also an insect that produces honey.
- Every bee is an insect and it also produces honey.
- A cow is a mammal that eats grass.

- A cow is a mammal that produces milk.

Many meanings, one structure.[2]

[2] Other semantic phenomena are outside the scope of a syntactic transformation approach.

First Challenge: the Dataset

Desiderata for the dataset:[3]

- covers many syntactic constructs (structure);
- covers many domains (meaning);
- has annotated <sentence, axiom> pairs.

[3] G. Petrucci. *"Information Extraction for Learning Expressive Ontologies"*, ESWC 2015 Ph.D. Symp.

First Challenge: the Dataset

Desiderata for the dataset:[3]

- covers many syntactic constructs (structure);
- covers many domains (meaning);
- has annotated $<$sentence, axiom$>$ pairs.

List of suitable datasets:

[3] G. Petrucci. *"Information Extraction for Learning Expressive Ontologies"*, ESWC 2015 Ph.D. Symp.

First Challenge: the Dataset

Desiderata for the dataset:[3]

- covers many syntactic constructs (structure);
- covers many domains (meaning);
- has annotated <sentence, axiom> pairs.

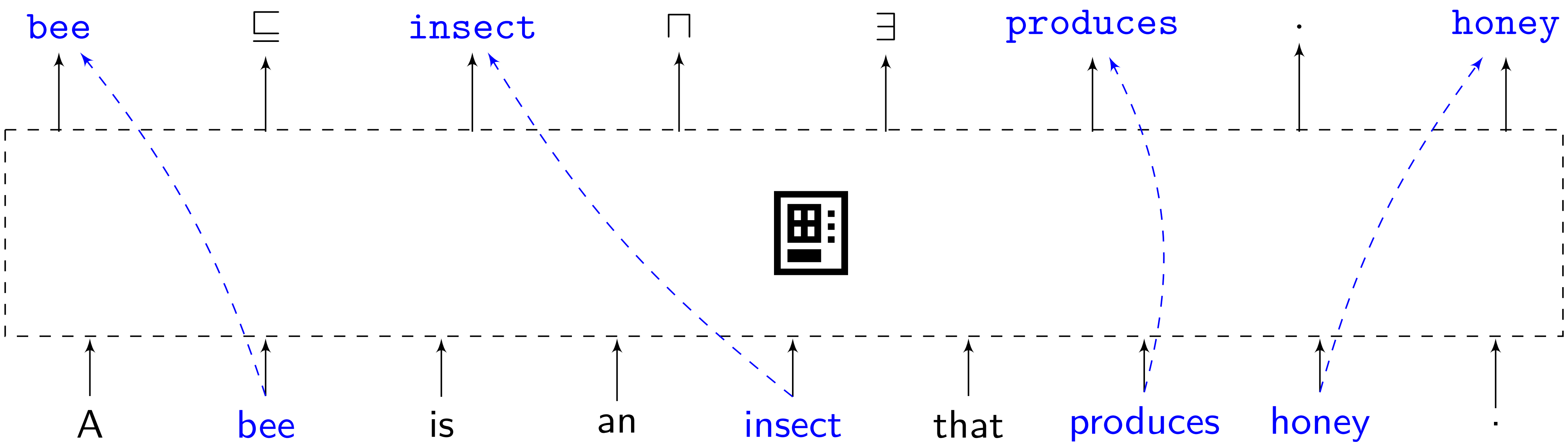
List of suitable datasets:

- ?

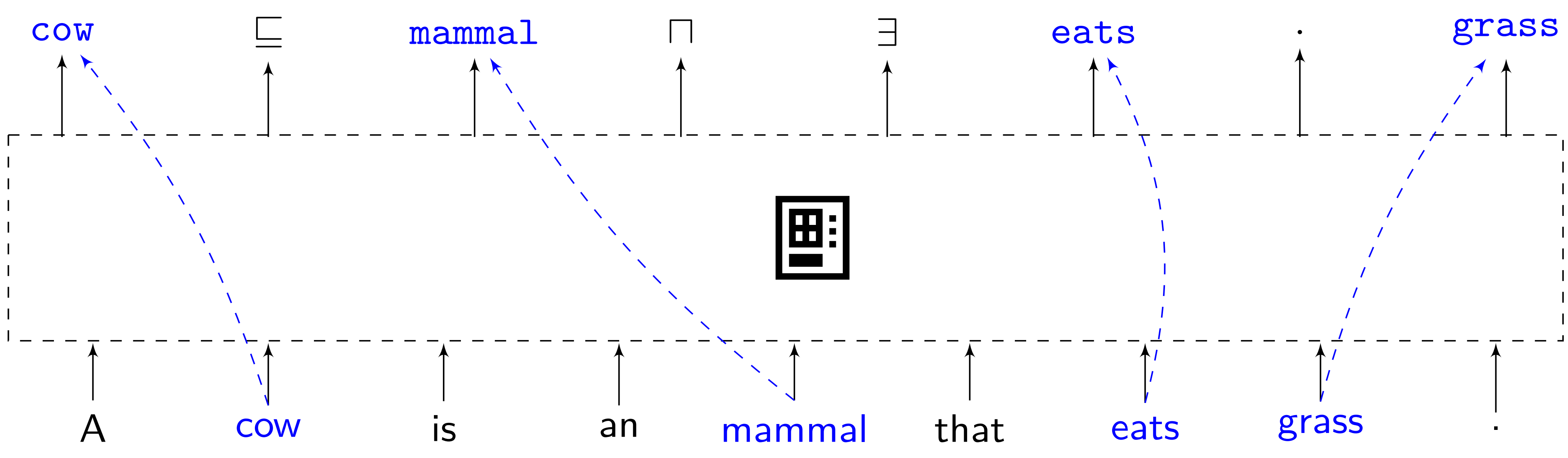
Following other notable approaches in literature, we started building a dataset to train our model.

[3] G. Petrucci. *"Information Extraction for Learning Expressive Ontologies"*, ESWC 2015 Ph.D. Symp.

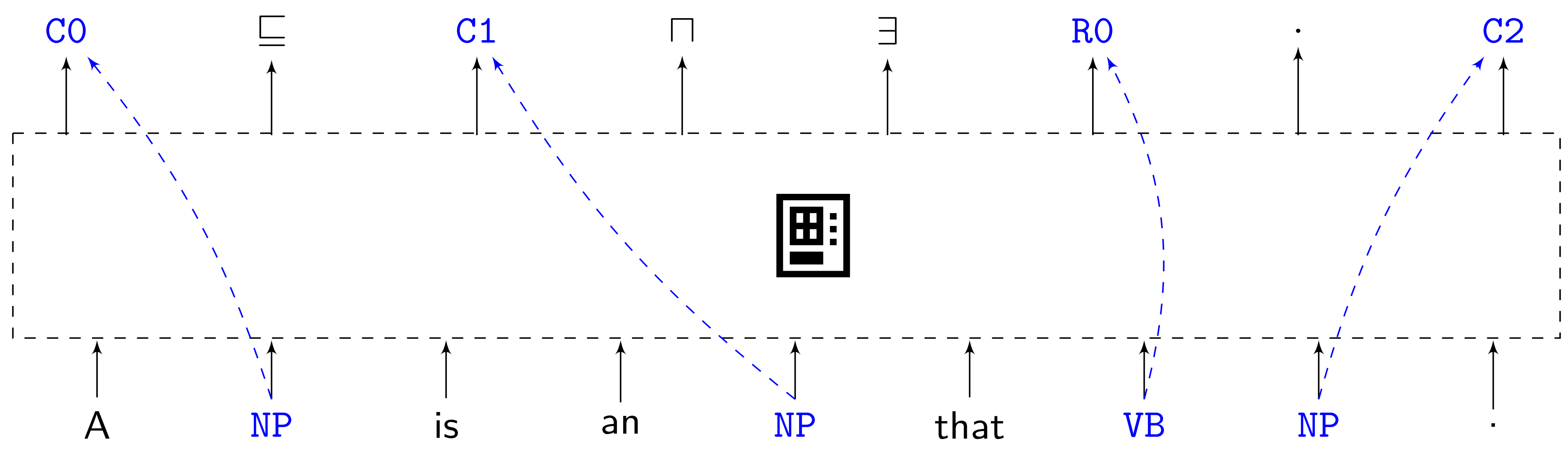
First Challenge: the Dataset



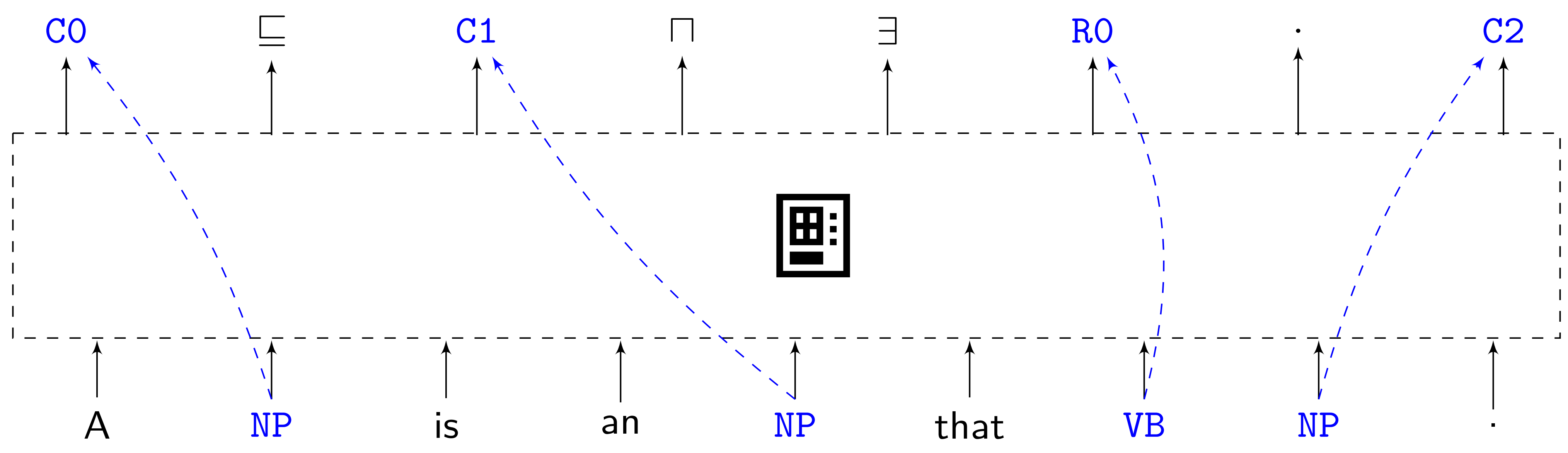
First Challenge: the Dataset



First Challenge: the Dataset



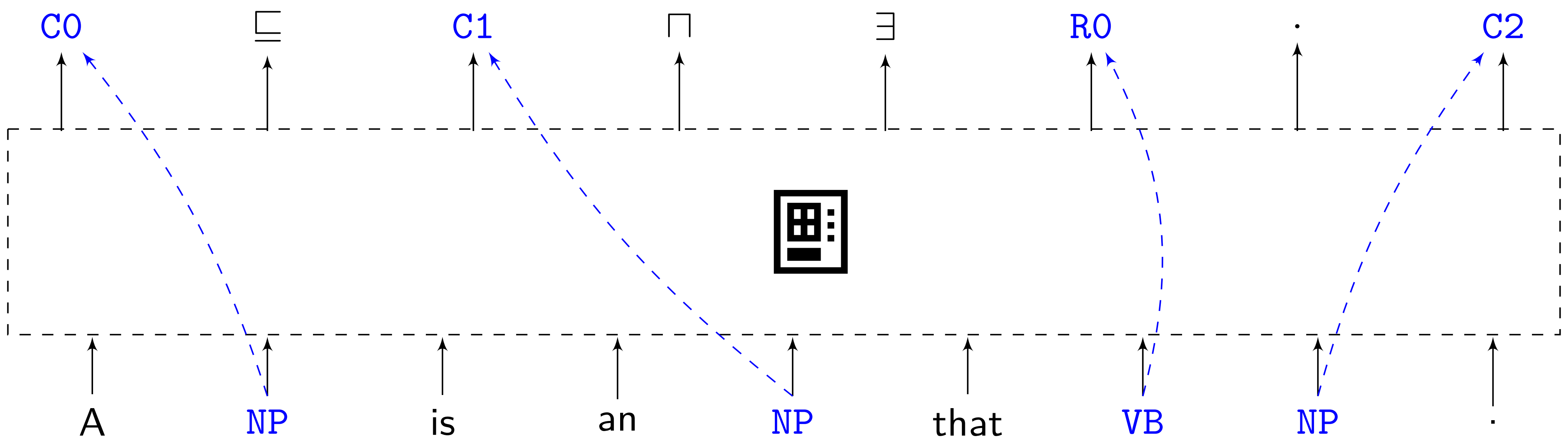
First Challenge: the Dataset



A NP is a NP that VB NP

$$\mathtt{C0} \sqsubseteq \mathtt{C1} \sqcap \exists \mathtt{R0}.\mathtt{C2}$$

First Challenge: the Dataset



A NP is a NP that VB NP

$$\mathtt{C0} \sqsubseteq \mathtt{C1} \sqcap \exists \mathtt{R0}.\mathtt{C2}$$

Templates: structural regularities beyond meaning.

Data Generation Process

Context-Free Grammar

Every `C0` `R0` at least `NUM` `C1`

Data Generation Process

Context-Free Grammar

↓

Every `C0 R0` at least `NUM C1`

$\mathtt{C0} \sqsubseteq \geqslant \mathtt{NUM}\ \mathtt{R0.C1}$

Data Generation Process

V

Context-Free Grammar

Every `C0` `R0` at least `NUM` `C1`

$\texttt{C0} \sqsubseteq \geqslant \texttt{NUM R0.C1}$

Every PhD student defends at least 1 thesis.

$\texttt{PhD student} \sqsubseteq \geqslant 1\ \texttt{defends.thesis}$

Data Generation Process

V

Context-Free Grammar

random!

Every `C0` `R0` at least `NUM` `C1`

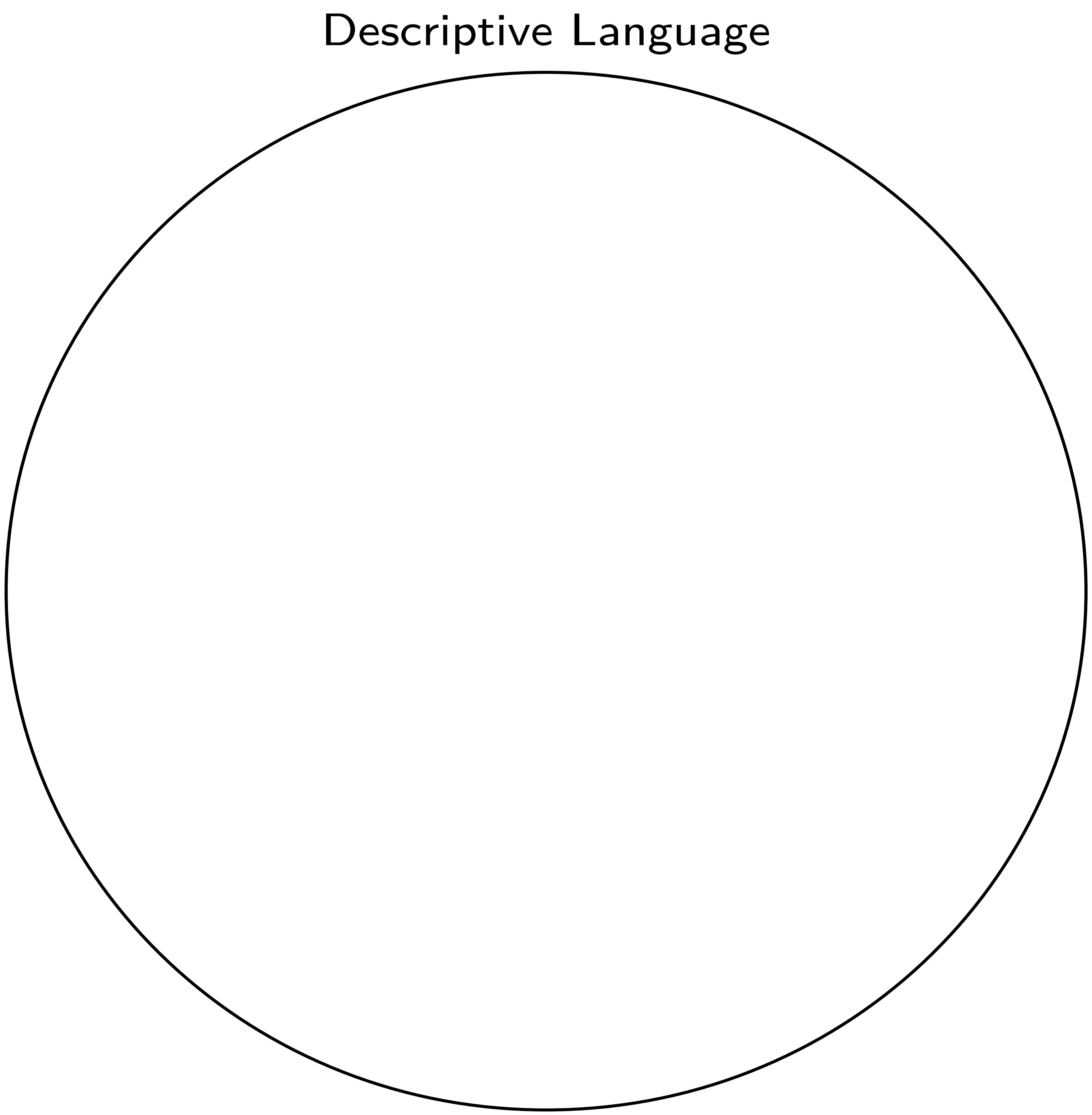
$\texttt{C0} \sqsubseteq\ \geqslant \texttt{NUM}\ \texttt{R0.C1}$

Every innocent exile craves at least 100 towers.

$\texttt{innocent exile} \sqsubseteq\ \geqslant 100\ \texttt{craves.towers}$

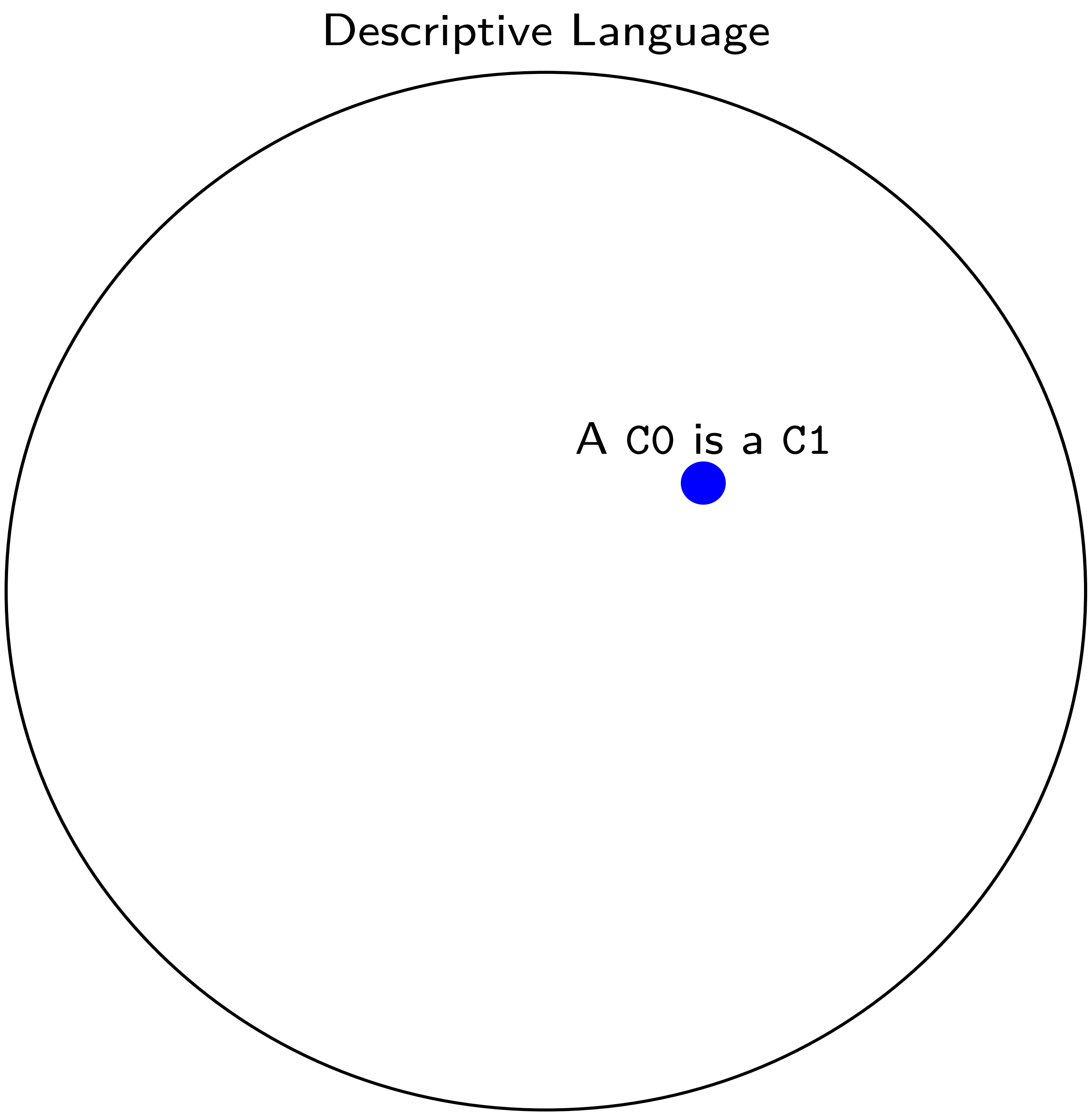
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



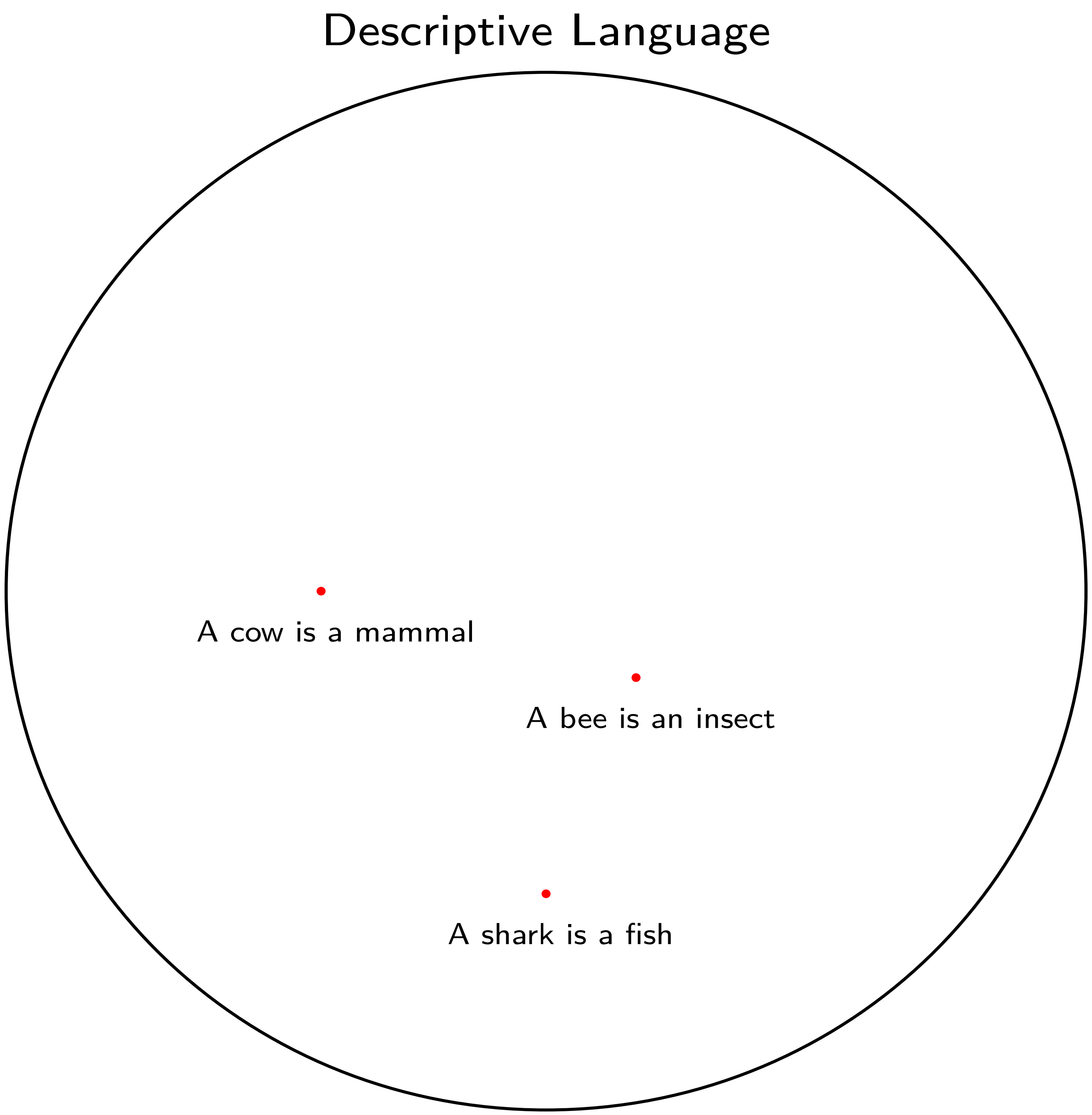
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



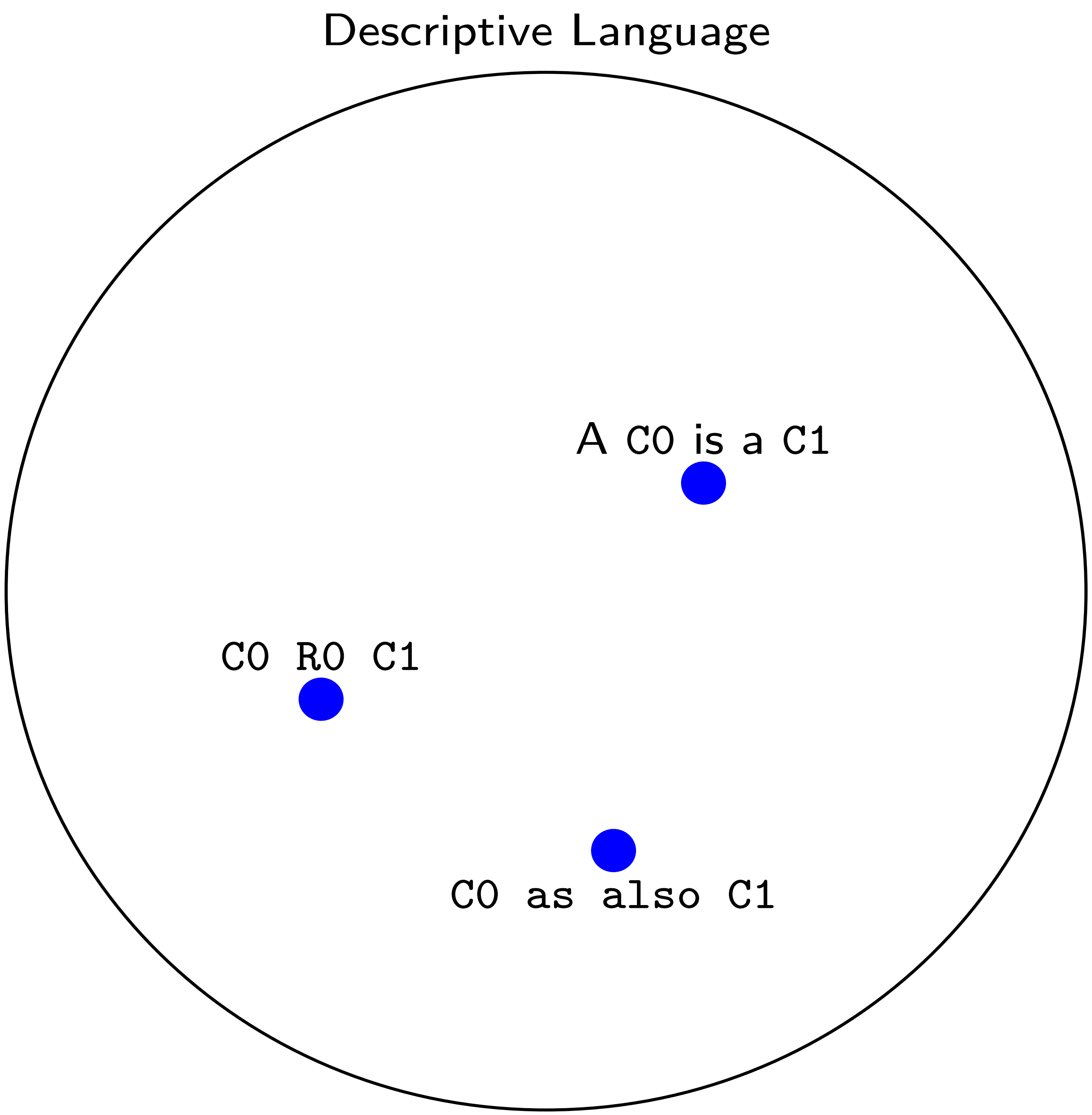
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



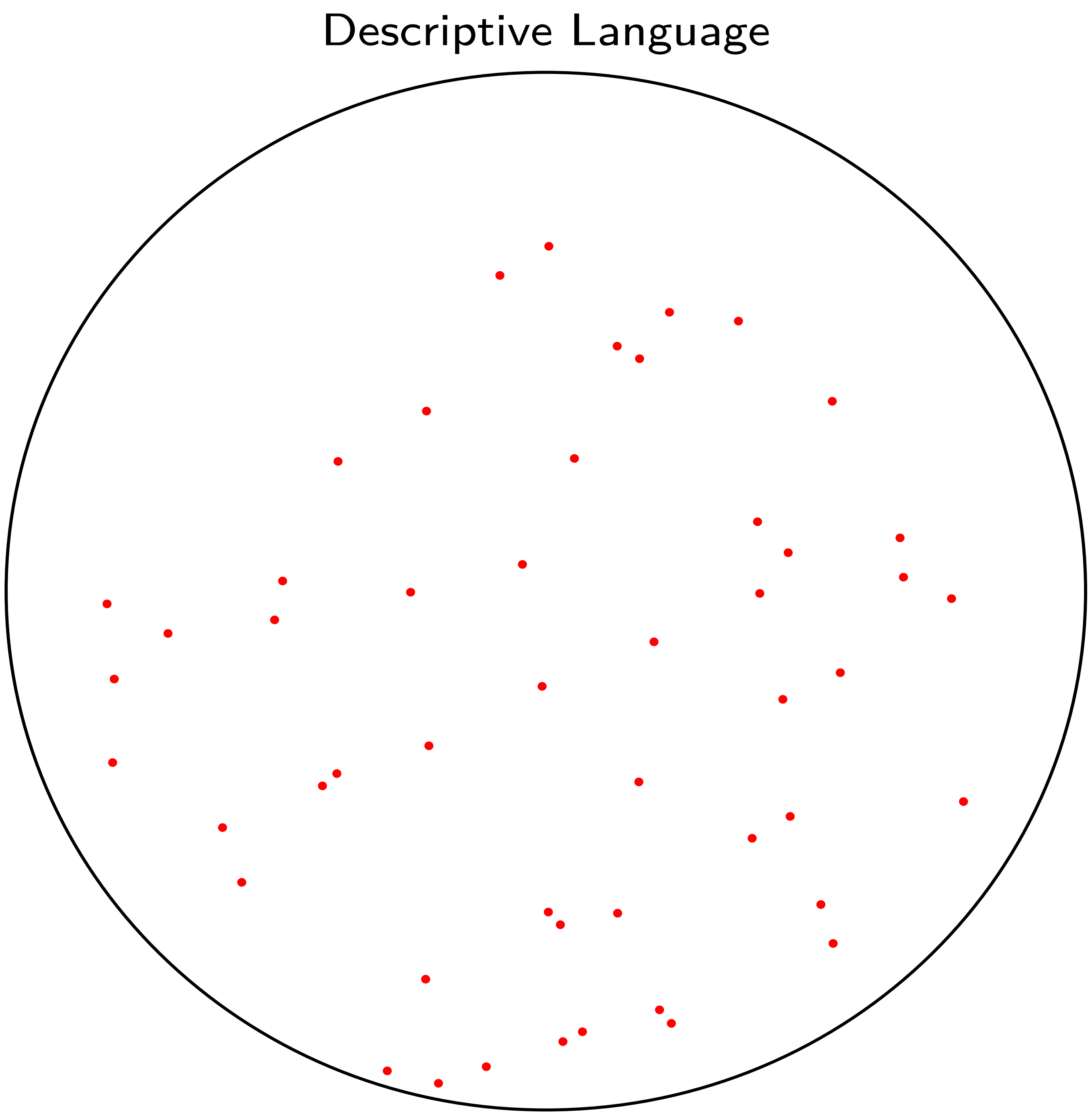
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



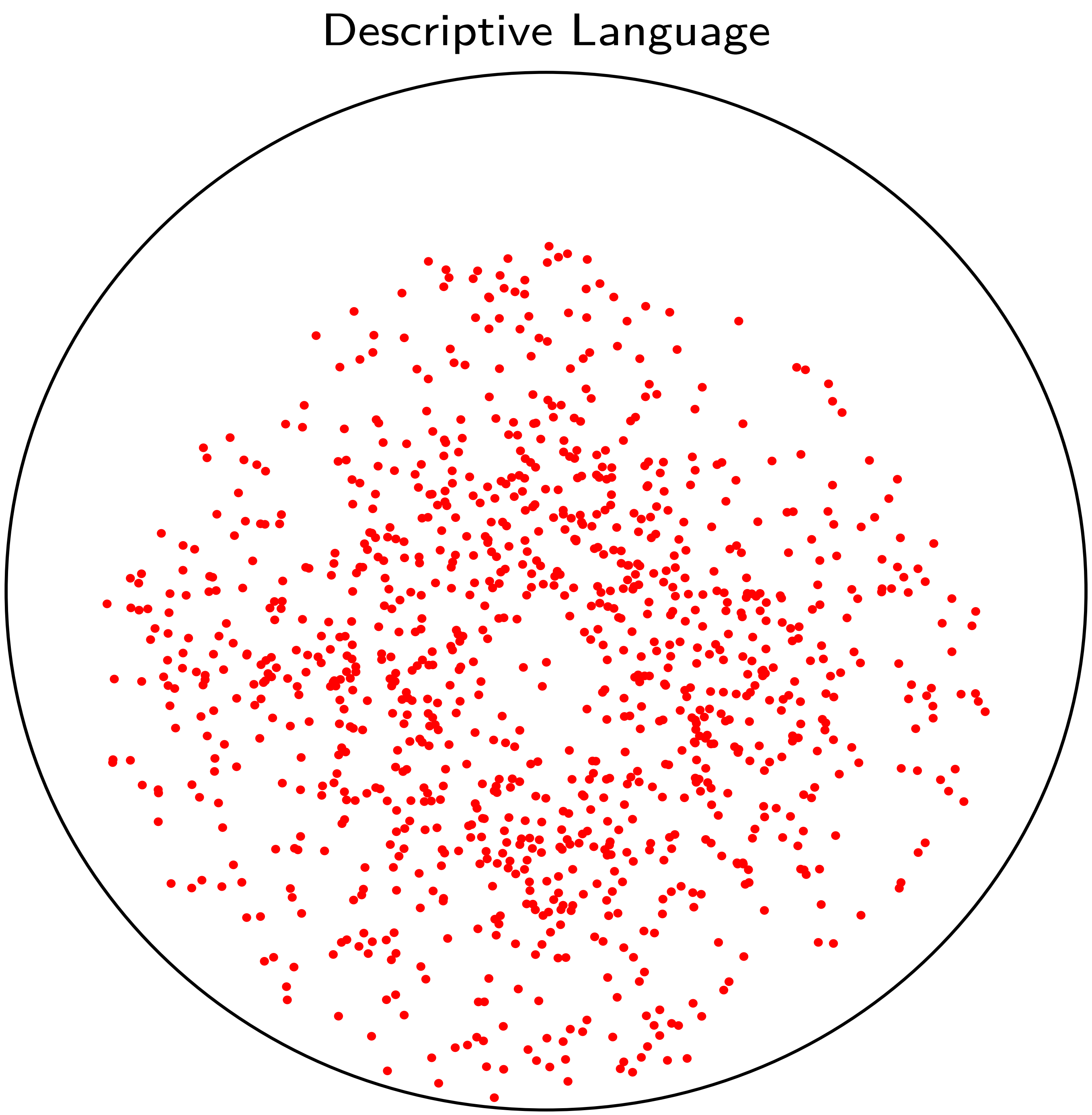
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



Data Generation Process

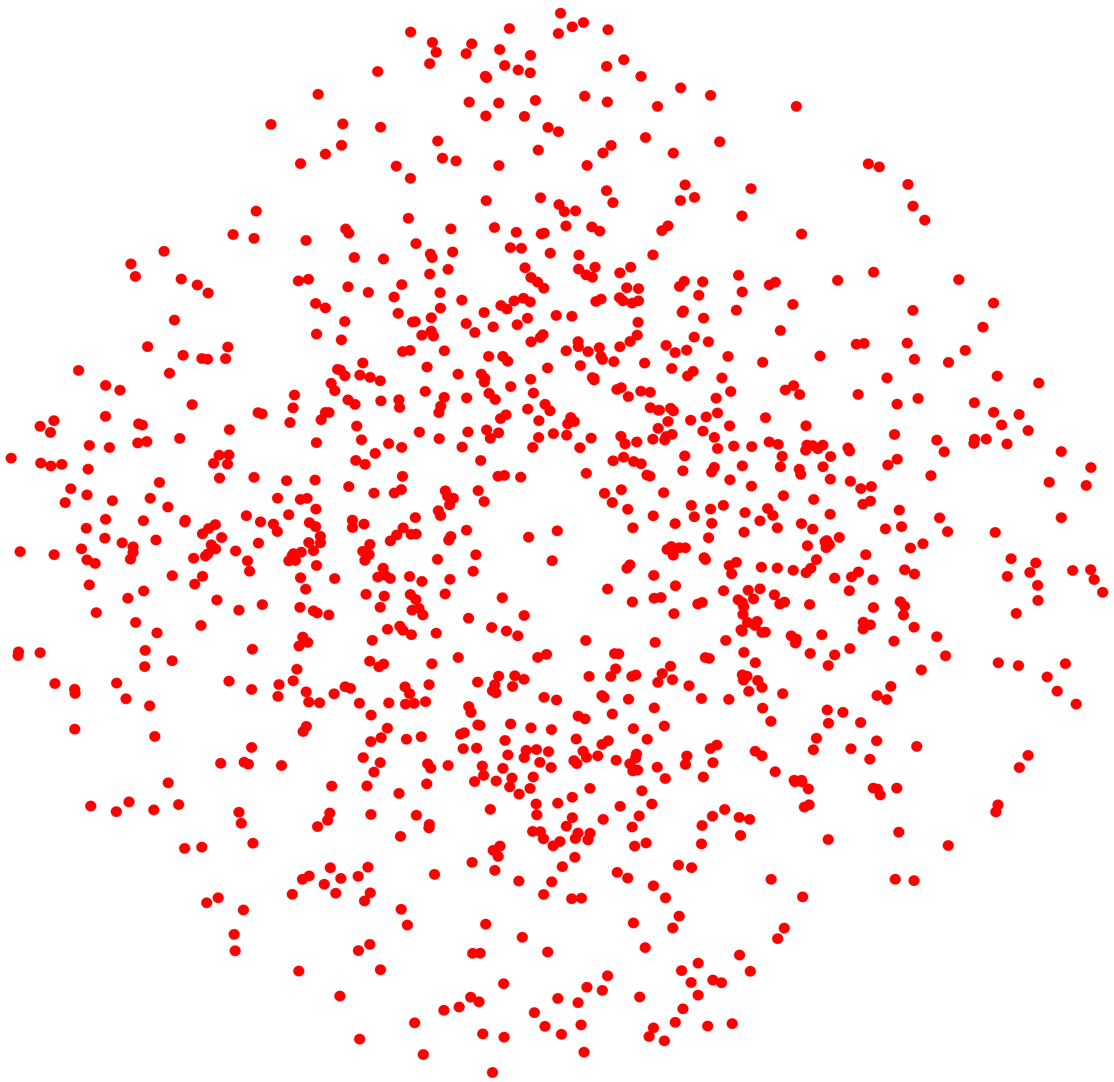
- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



Data Generation Process

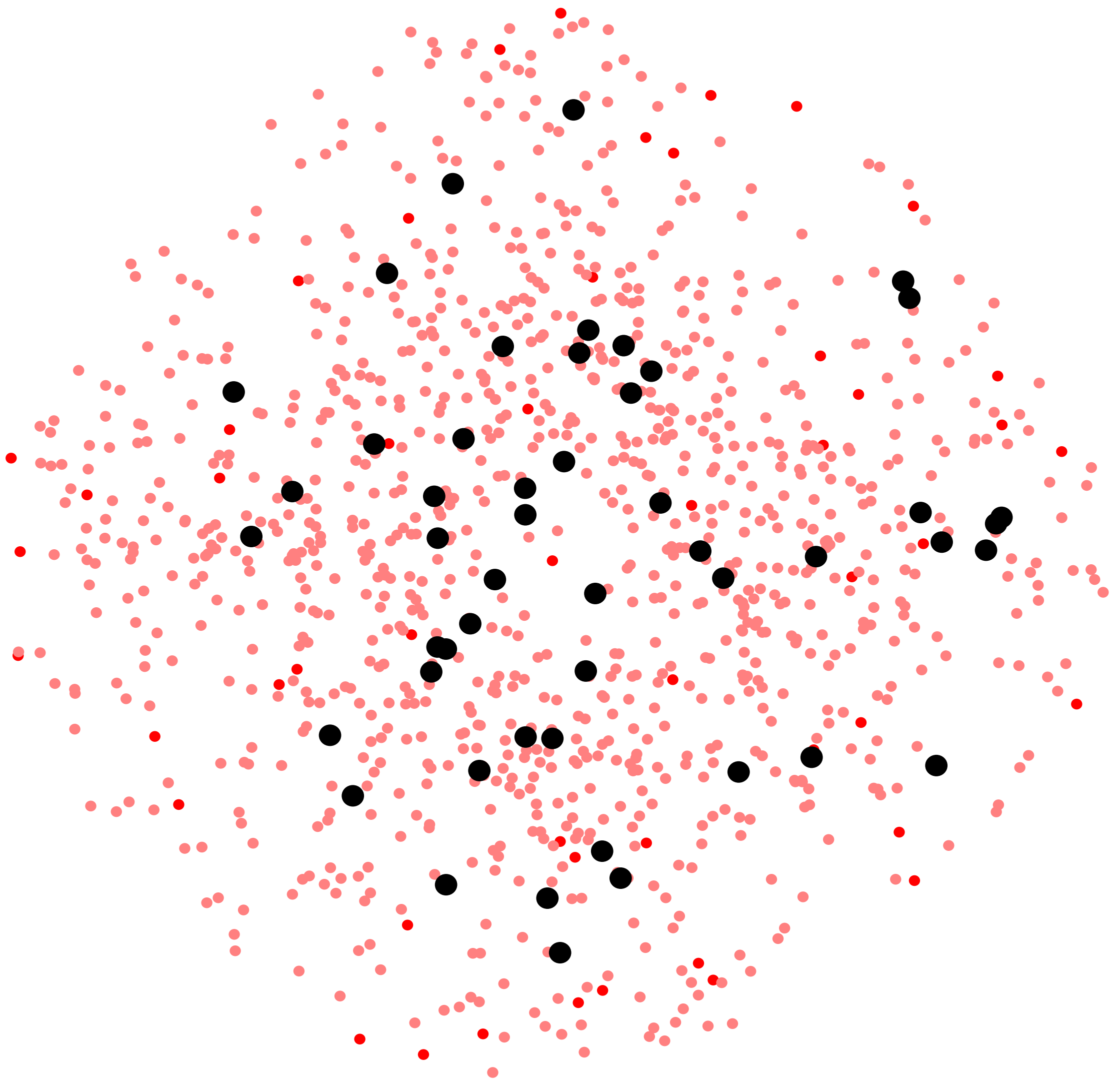
- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.

$\sim$ Descriptive Language



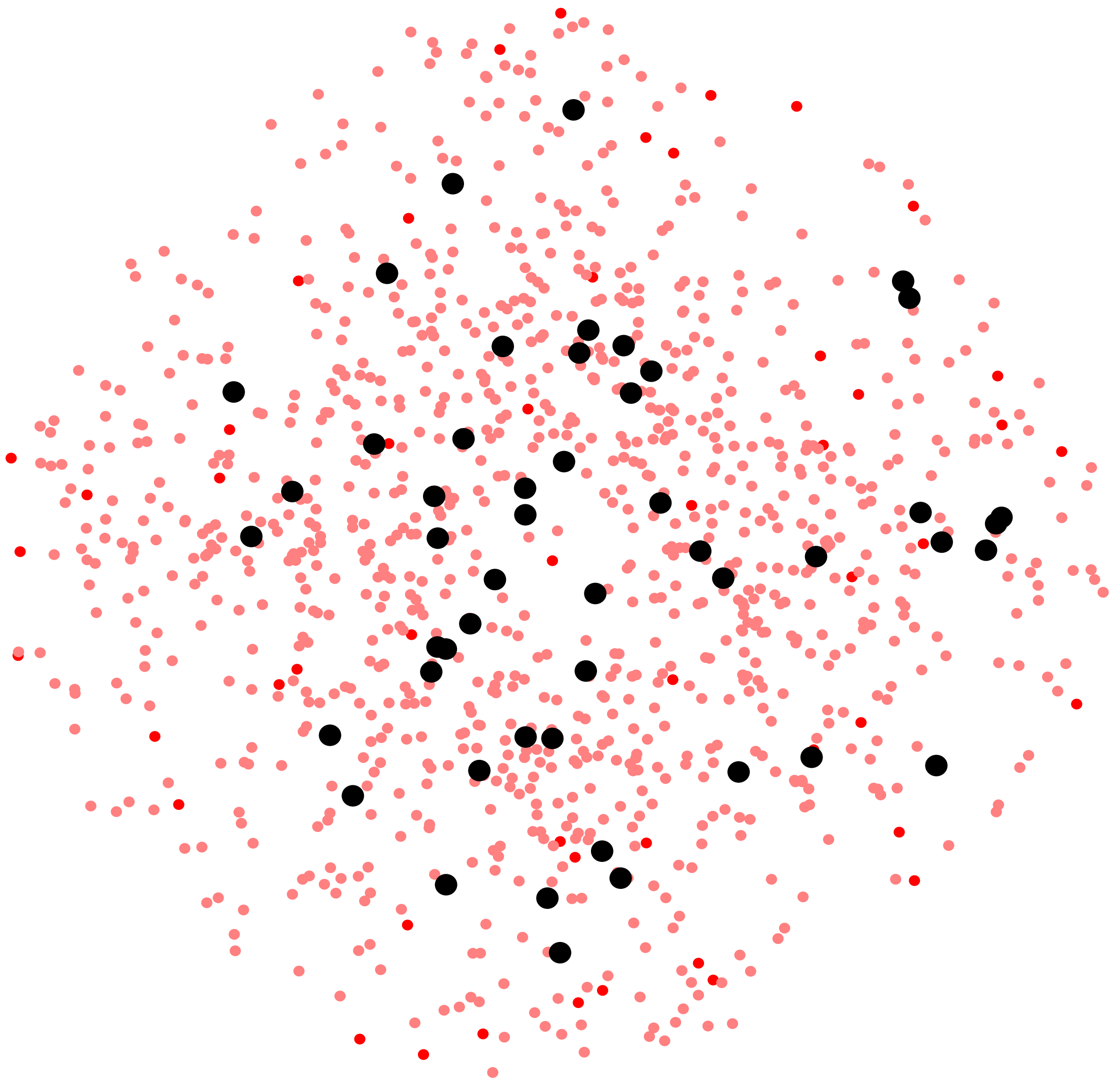
Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



Data Generation Process

- Context-Free Grammar
- template;
- actualization;
- (repeat);
- approximation;
- sampling...
- and parsing.



Data Generation Process

Minimal input preprocessing:

- lower-cased;
- *"an"* $\rightarrow$ *"a"*;
- *"doesn't"*, *"does not"*, *"don't"* $\rightarrow$ *"do not"*;
- lemmatised nouns and verbs;
- numbers $\rightarrow$ `NUM`;

Second Challenge: the Model

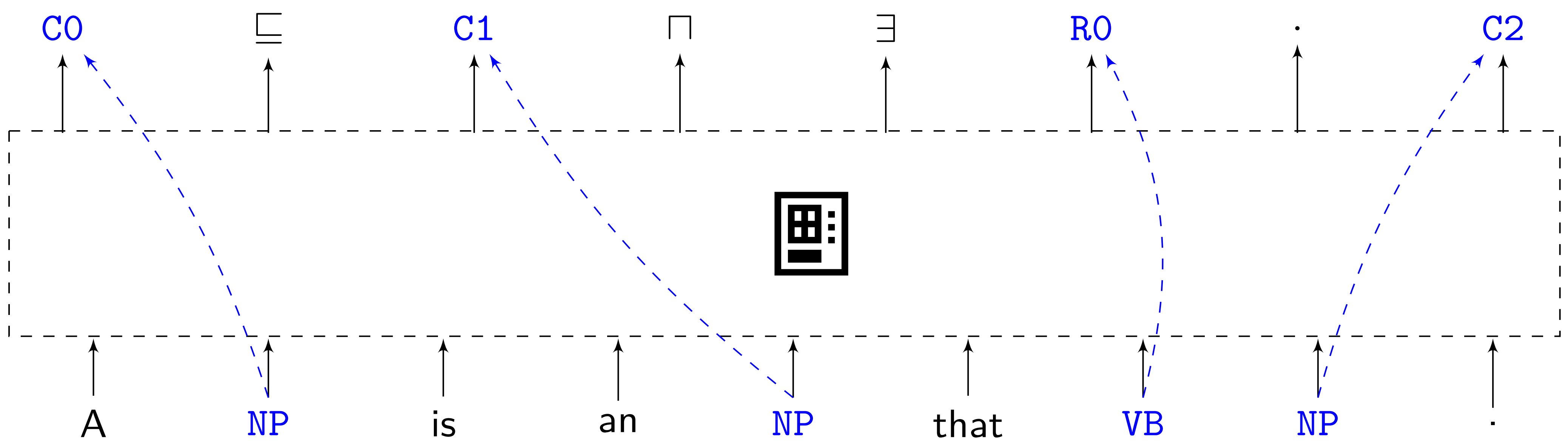
A 2-chapter adventure in the world of Recurrent Neural Networks:

2014-2015 Tag&Transduce
G. Petrucci, C. Ghidini, and M. Rospocher
"Ontology Learning in the Deep"
EKAW 2016

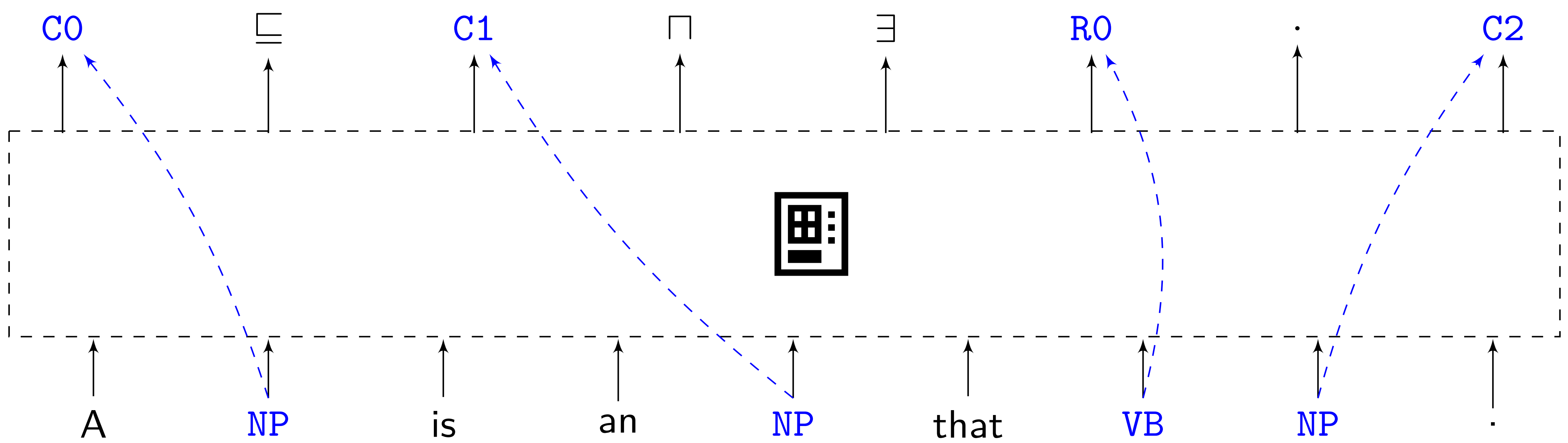
2016-2017 Translate
G. Petrucci, M. Rospocher, and C. Ghidini
"Expressive Ontology Learning as Neural Machine Translation Task"
(Under review)[4]

[4] Code & Datasets: `https://github.com/dkmfbk/dket`

Tagging and Transducing

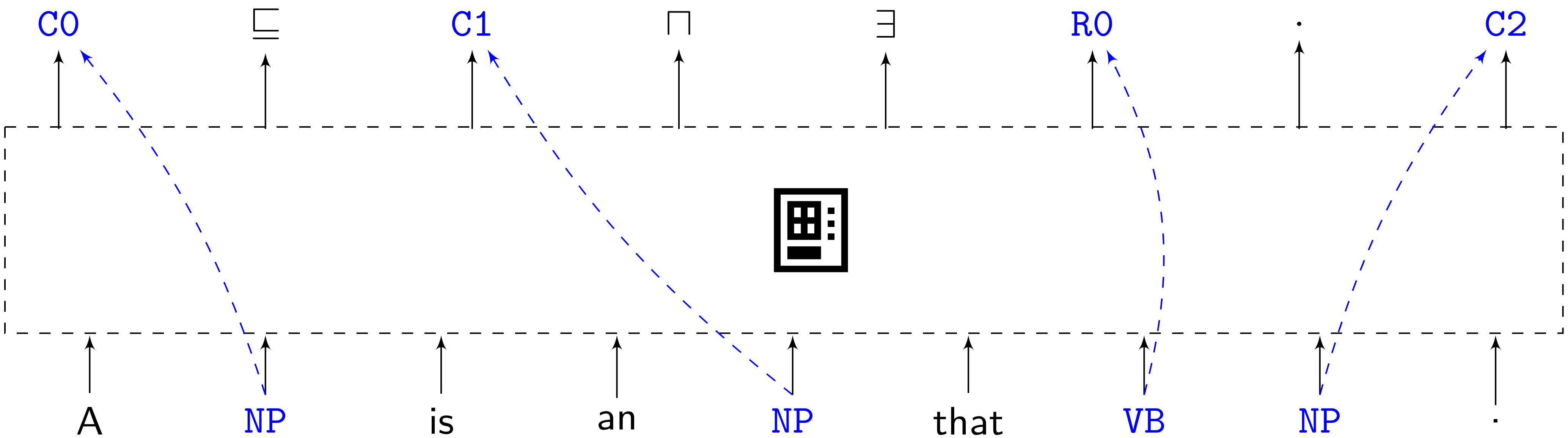


Tagging and Transducing



Transduction from sentence to formula template;

Tagging and Transducing



Transduction from sentence to formula template;

Tagging extralogical symbols at the right place;

Tagging and Transducing

A bee is an insect that produces honey.

Tagging and Transducing

$$\texttt{C0} \sqsubseteq \texttt{C1} \sqcap \exists\, \texttt{R0.C2;}$$

transduction (F)

A bee is an in-sect that produces honey.

Tagging and Transducing

$\mathtt{C0} \sqsubseteq \mathtt{C1} \sqcap \exists\ \mathtt{R0.C2;}$

transduction (F)

?

A bee is an insect that produces honey.

$\mathtt{Bee} \sqsubseteq \mathtt{Insect} \sqcap \exists\, \mathtt{produces.Honey}$

Tagging and Transducing

$\mathtt{C0} \sqsubseteq \mathtt{C1} \sqcap \exists\ \mathtt{R0.C2};$

transduction (F)

?

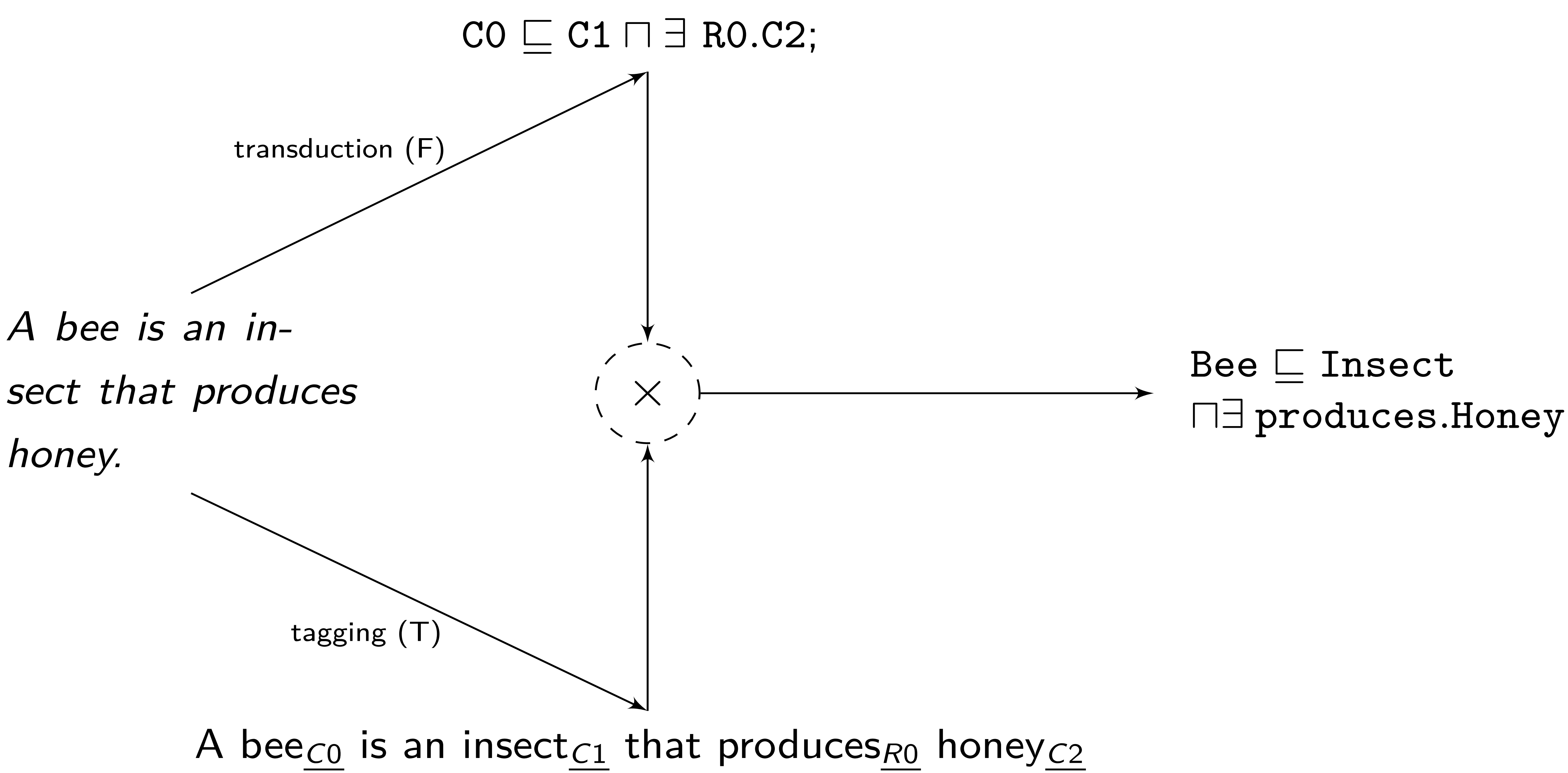
A bee is an insect that produces honey.

$\mathtt{Bee} \sqsubseteq \mathtt{Insect}$
$\sqcap \exists\, \mathtt{produces.Honey}$

tagging (T)

A bee$_{\underline{C0}}$ is an insect$_{\underline{C1}}$ that produces$_{\underline{R0}}$ honey$_{\underline{C2}}$

Tagging and Transducing



The Transducing Network

PREDICTION
$y_k = argmax(\mathbf{y}_k)$

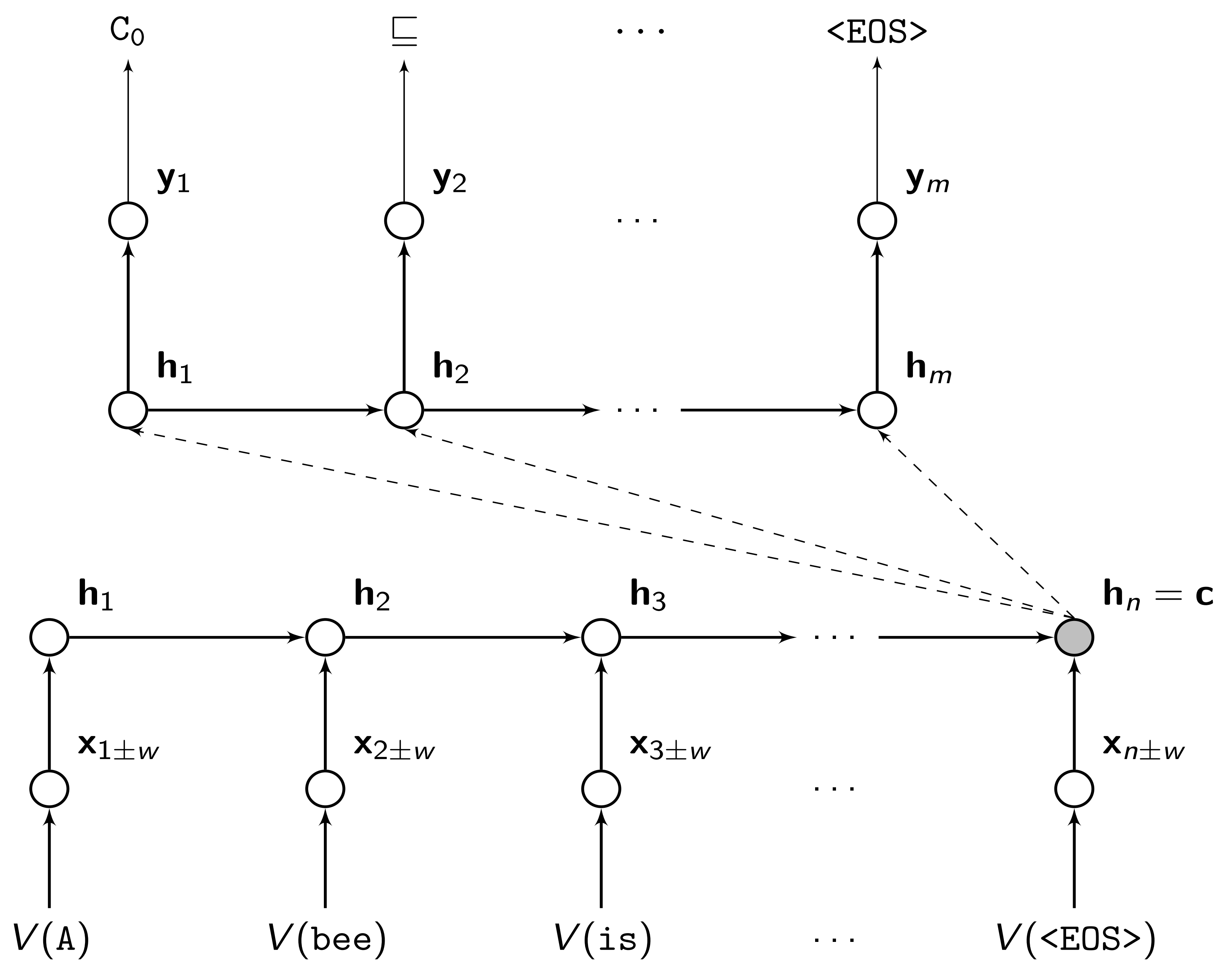
OUTPUT
$\mathbf{y}_j = softmax(\mathbf{W}\mathbf{h}_j + \mathbf{b})$

DECODING
$\mathbf{h}_j = g(\mathbf{c}, \mathbf{h}_{j-1})$

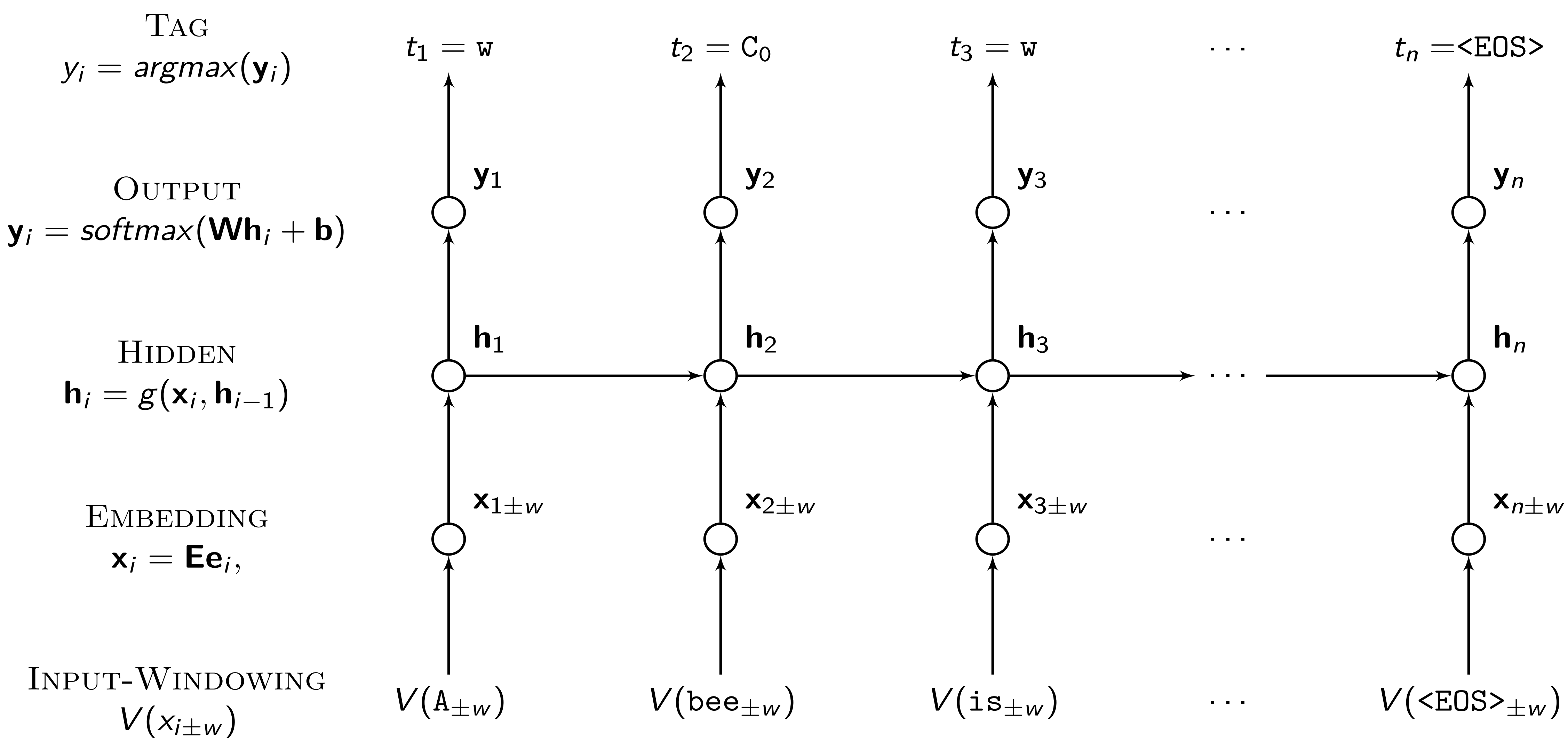
HIDDEN
$\mathbf{h}_i = g(\mathbf{x}_i, \mathbf{h}_{i-1})$

EMBEDDING
$\mathbf{x}_i = \mathbf{E}e_i,$

INPUT
$V(x_i)$



The Tagging Network



Tagging and Transducing: Evaluation

RQ1. To what degree is the network capable to generalize over the syntactic structures of descriptive language? (many structures, one meaning)

RQ2. To what degree is the network capable to tolerate words that have not been seen during the training phase? (many meanings, one structure)

Tagging and Transducing: Evaluation

Evaluation Metrics:

Avg. Per-Formula Acc. $FA(\hat{\mathcal{F}}, \mathcal{F}) = \frac{CF}{M} = \frac{\sum_{k=1}^{M} \begin{cases} 1, & \text{if } f^k \equiv \hat{f}^k \\ 0, & \text{otherwise} \end{cases}}{M}$ fully automated

Avg. Edit Distance $ED(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \delta(f^k, \hat{f}^k)}{M}$ semi-automated

Avg. Per-Token Acc. $TA(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \sum_{j=1}^{T_{f^k}} \begin{cases} 1, & \text{if } f_j^k = \hat{f}_j^k \\ 0, & \text{otherwise} \end{cases}}{\sum_{k=1}^{M} T_{f^k}}$ quick control

Tagging and Transducing: Evaluation

- different training set sizes;
- 2M test examples;
- `<UNK>` between 20% and 40%.

Tagging and Transducing: Evaluation

- different training set sizes;
- 2M test examples;
- `<UNK>` between 20% and 40%.

training set size	**CF**	**FA**	**ED**	**TA**
1000	10	0.5e-5	2.67	0.90
2000	**161**	8.05e-5	1.34	0.95
3000	60	3.00e-5	1.22	0.96
4000	22	1.10e-5	**1.07**	**0.97**

Tagging and Transducing: Evaluation

- different training set sizes;
- 2M test examples;
- `<UNK>` between 20% and 40%.

training set size	CF	FA	ED	TA
1000	10	0.5e-5	2.67	0.90
2000	**161**	8.05e-5	1.34	0.95
3000	60	3.00e-5	1.22	0.96
4000	22	1.10e-5	**1.07**	**0.97**

Many limitations: we dropped the project and move forward.

Moving forward (aka $1 > 2$)

The placeholders are numbered in the training set and there is no way to overcome this limit—namely, generalize over the length of the sentence—by design.

Moving forward (aka 1 > 2)

Negramaro is a red and strong wine.

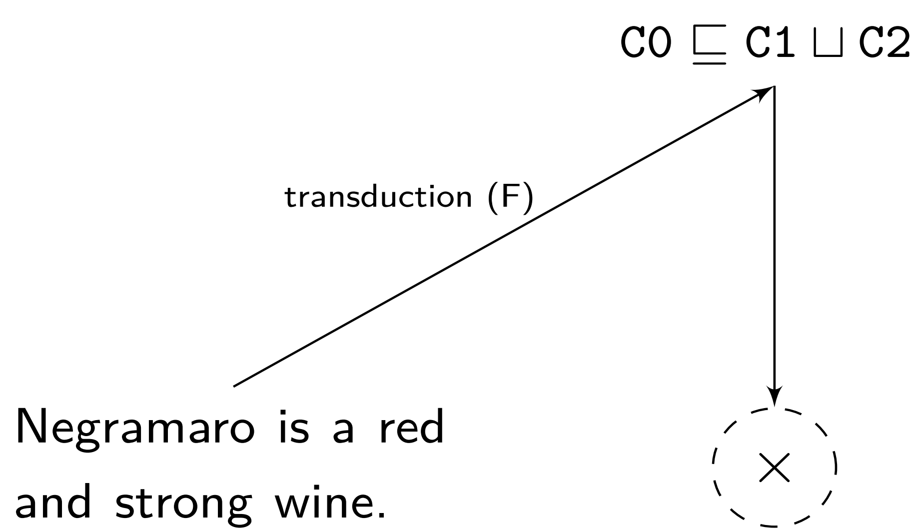
$$\texttt{negramaro} \sqsubseteq \texttt{red\_wine} \sqcap \texttt{strong\_wine}$$

Negramaro is a red
and strong wine.

Moving forward (aka 1 > 2)

Negramaro is a red and strong wine.

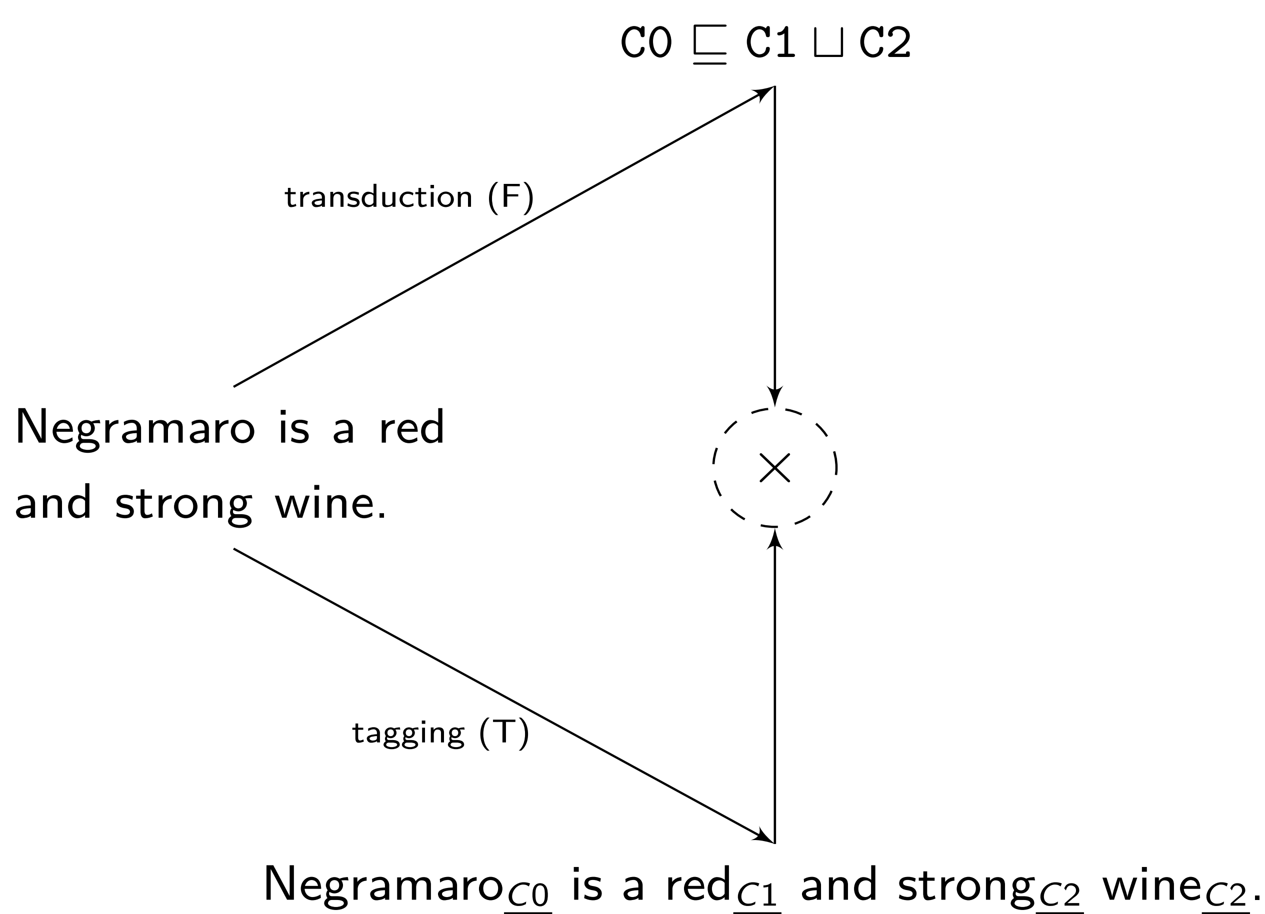
$\texttt{negramaro} \sqsubseteq \texttt{red\_wine} \sqcap \texttt{strong\_wine}$



Moving forward (aka 1 > 2)

Negramaro is a red and strong wine.

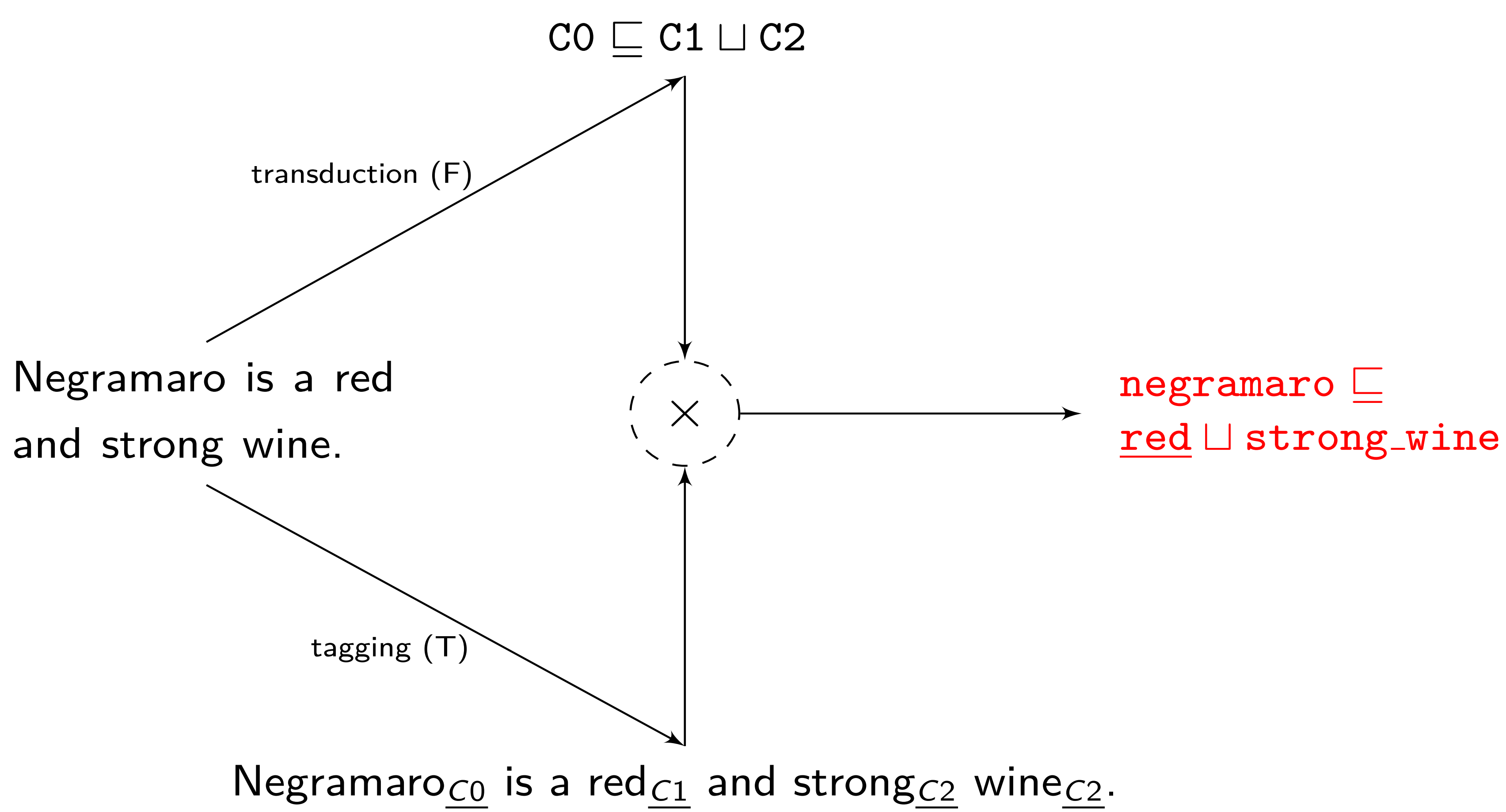
$$\texttt{negramaro} \sqsubseteq \texttt{red\_wine} \sqcap \texttt{strong\_wine}$$



Moving forward (aka 1 > 2)

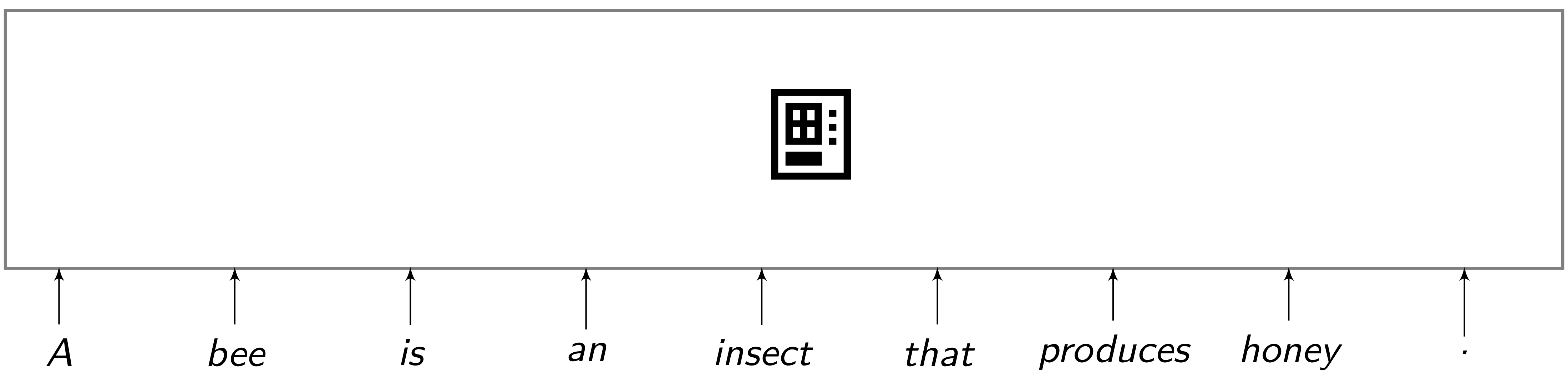
Negramaro is a red and strong wine.

$$\texttt{negramaro} \sqsubseteq \underline{\texttt{red\_wine}} \sqcap \texttt{strong\_wine}$$

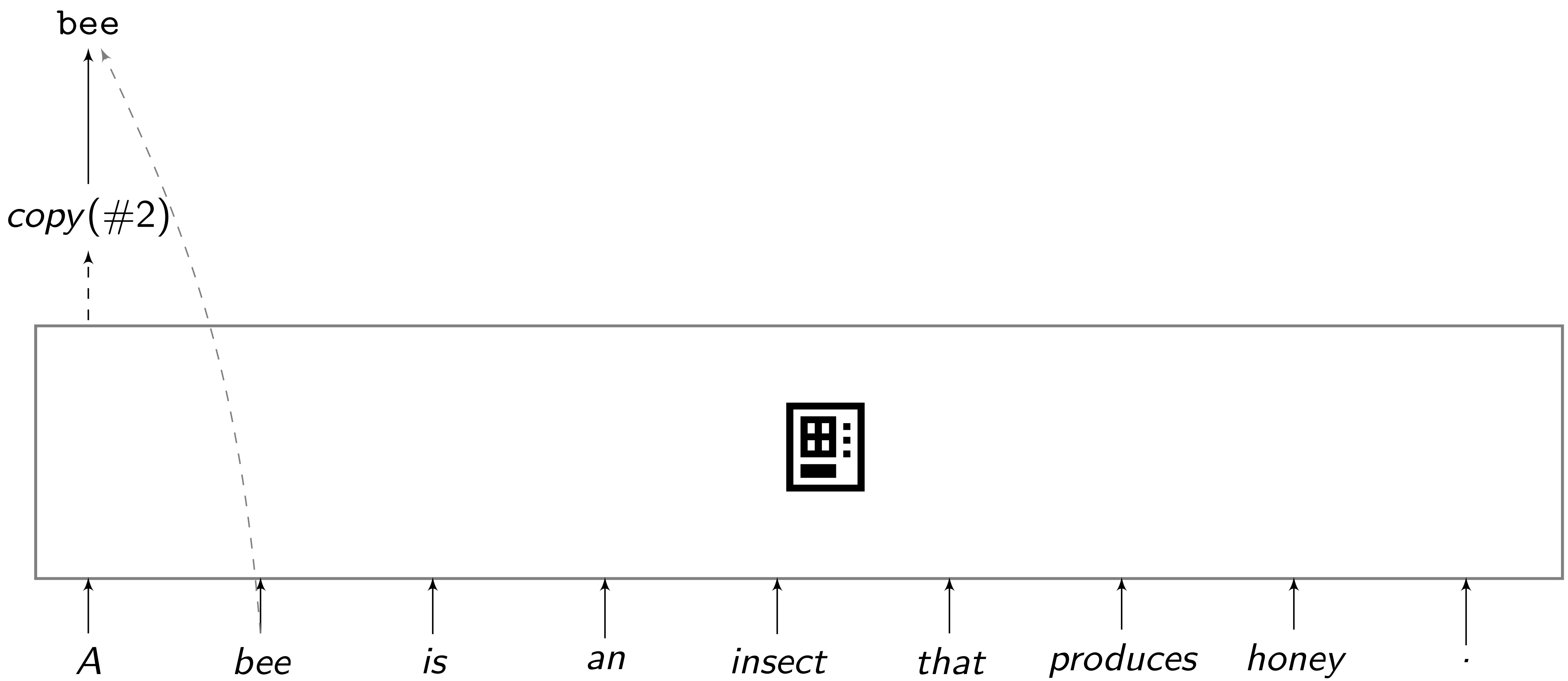


Translate

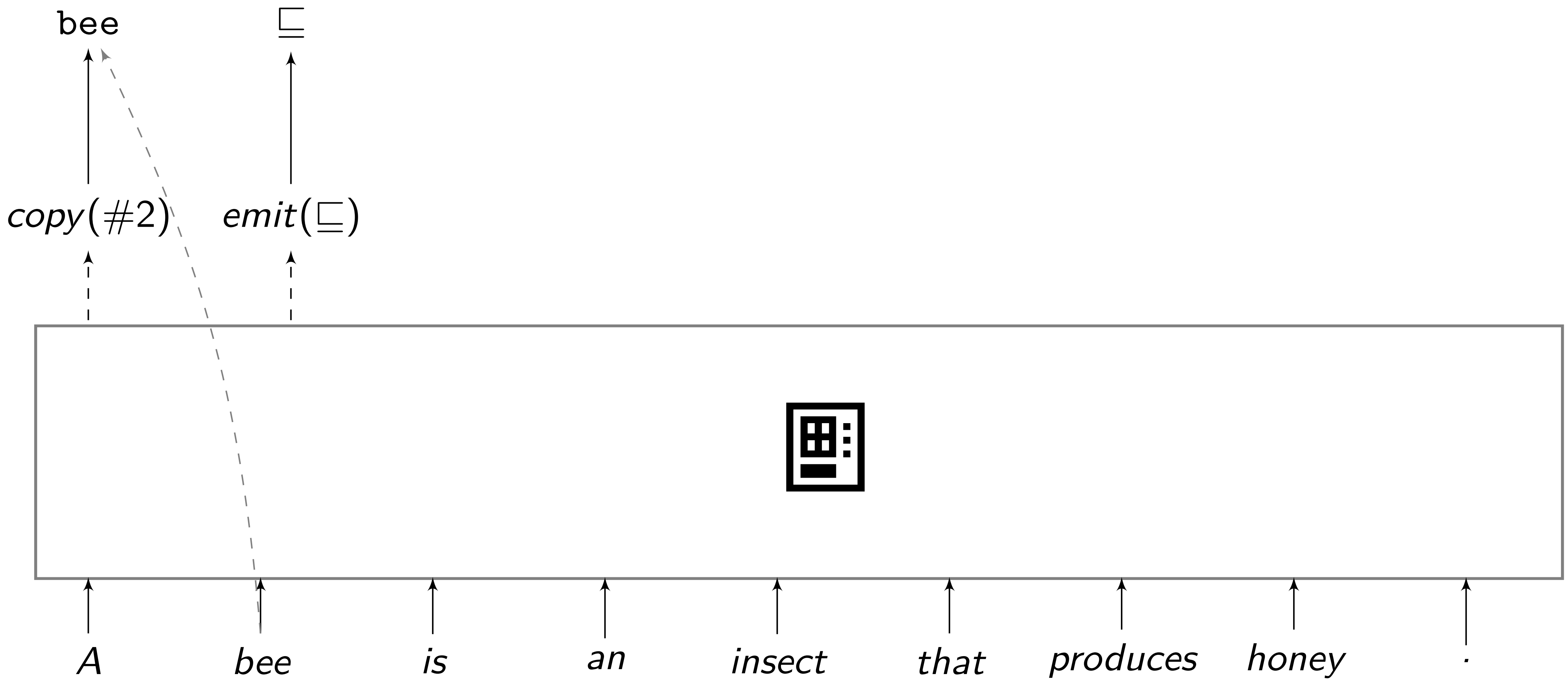
Translate



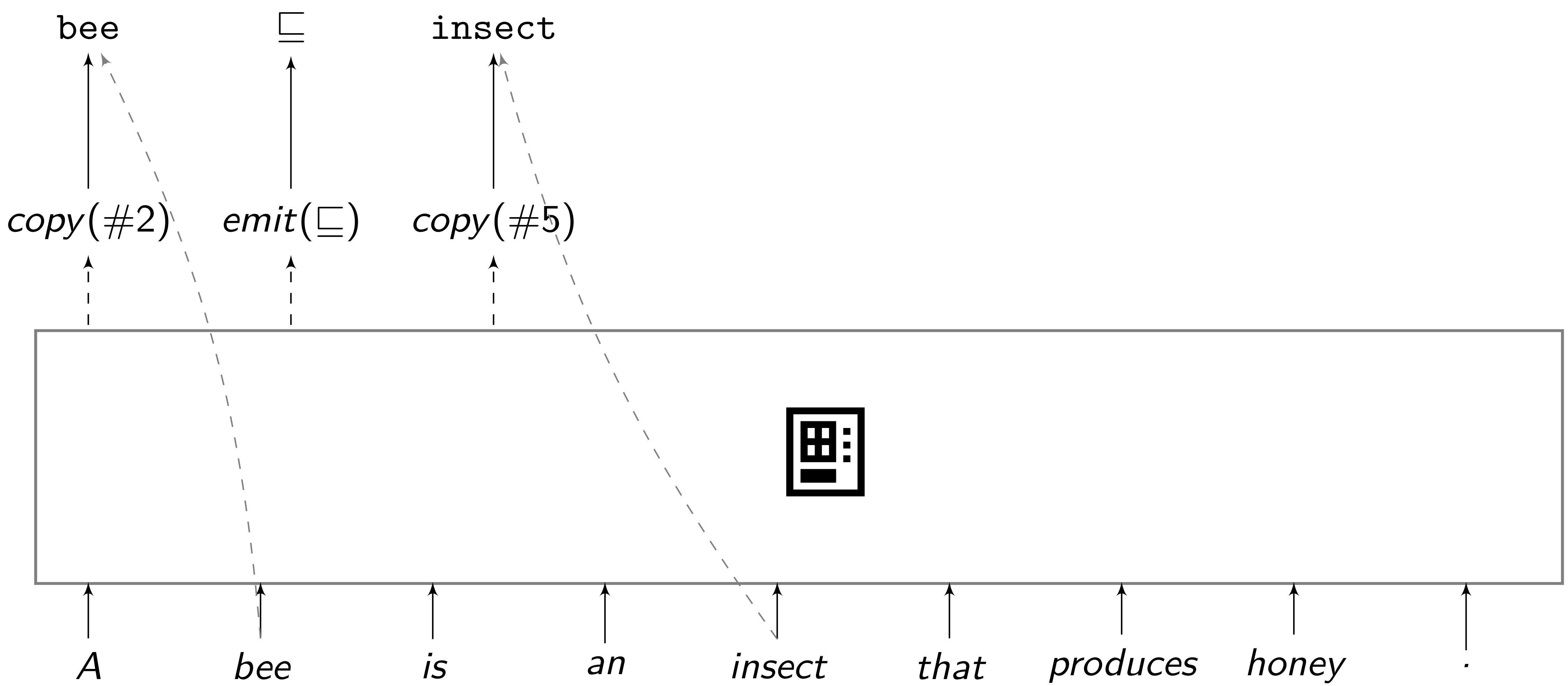
Translate



Translate

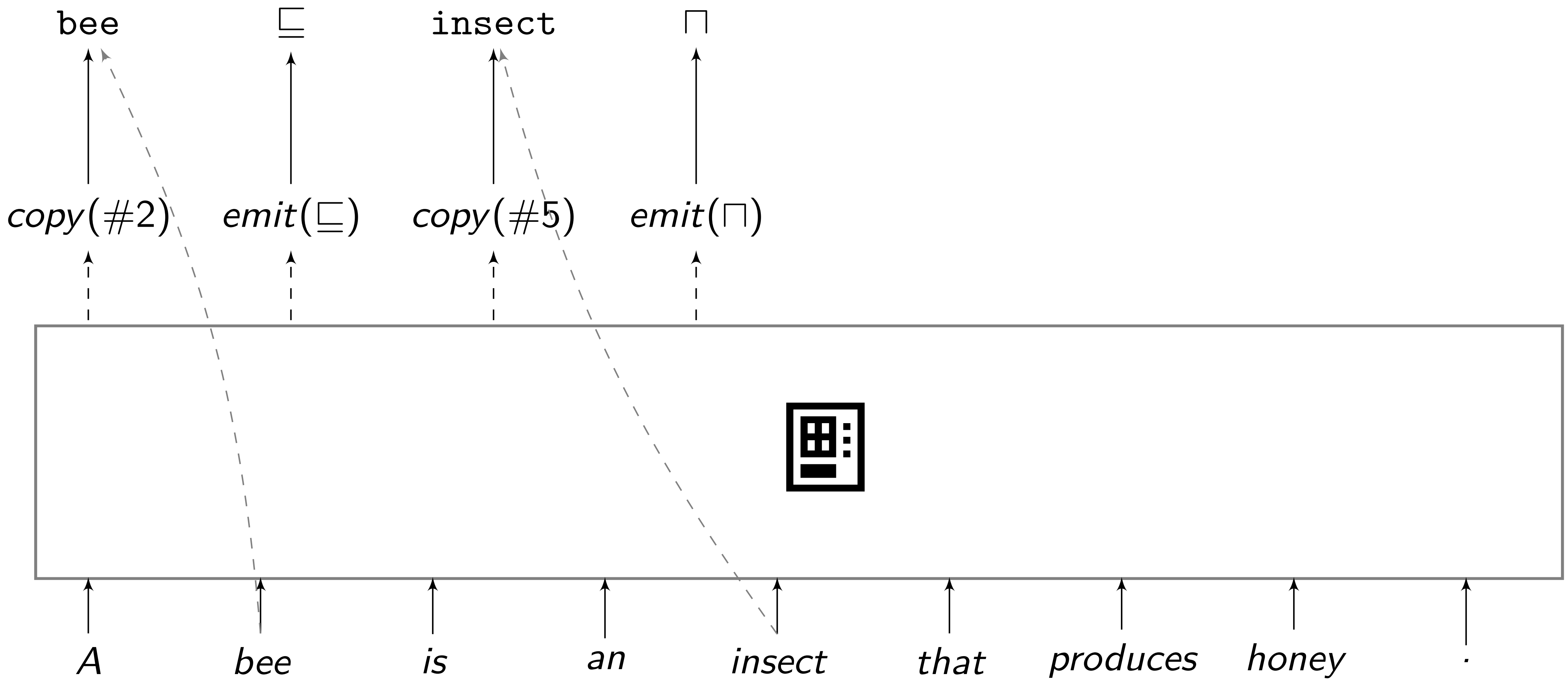


Translate



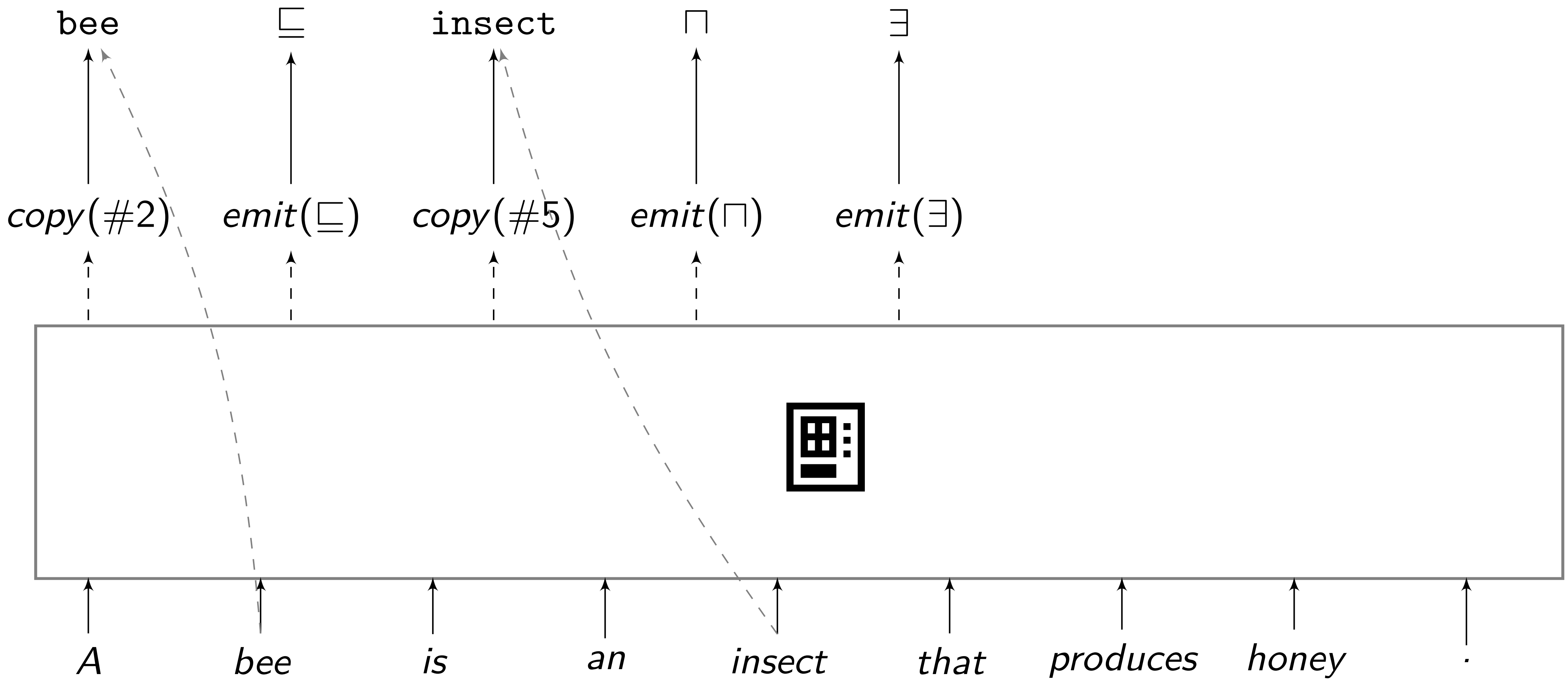
bee
⊑
insect
copy(#2)
emit(⊑)
copy(#5)
A
bee
is
an
insect
that
produces
honey
.

Translate



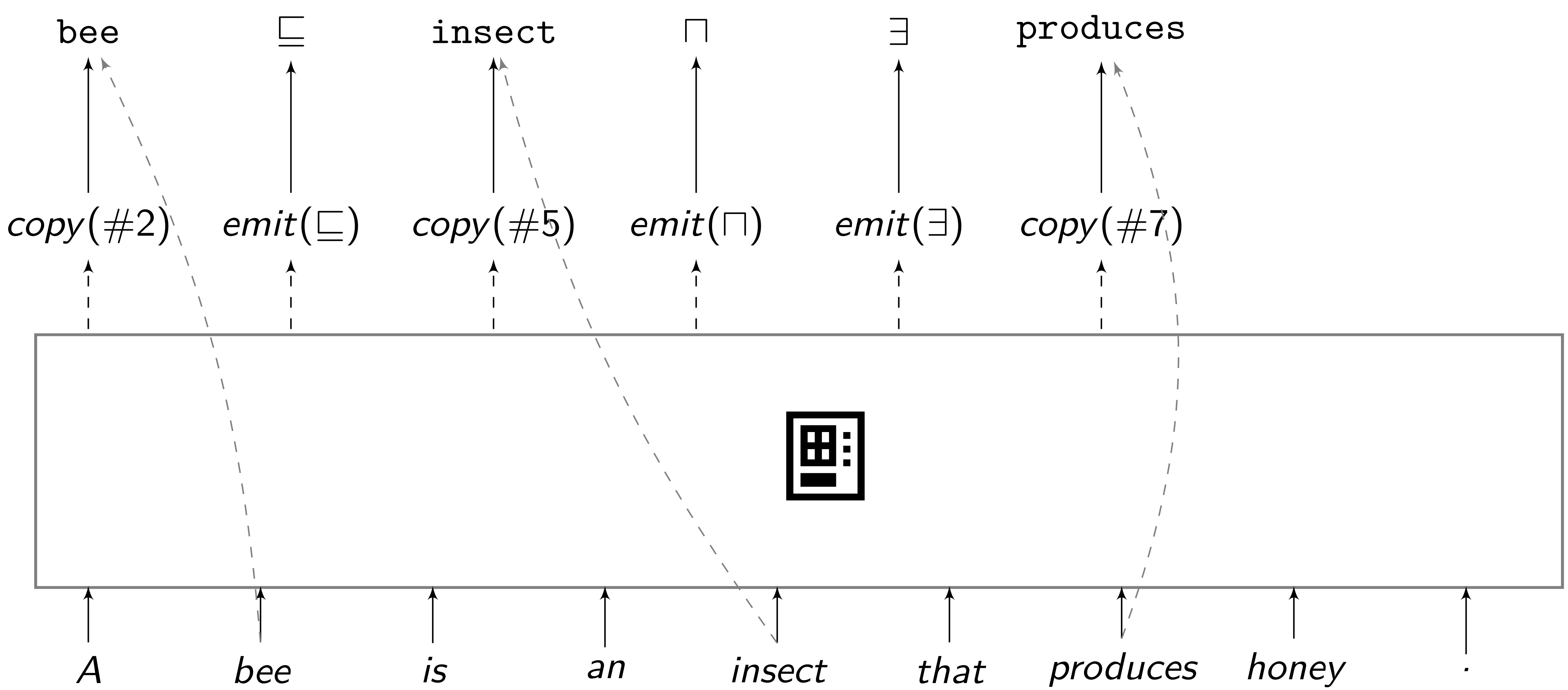
bee
⊑
insect
⊓
copy(#2)
emit(⊑)
copy(#5)
emit(⊓)
A
bee
is
an
insect
that
produces
honey
.

Translate

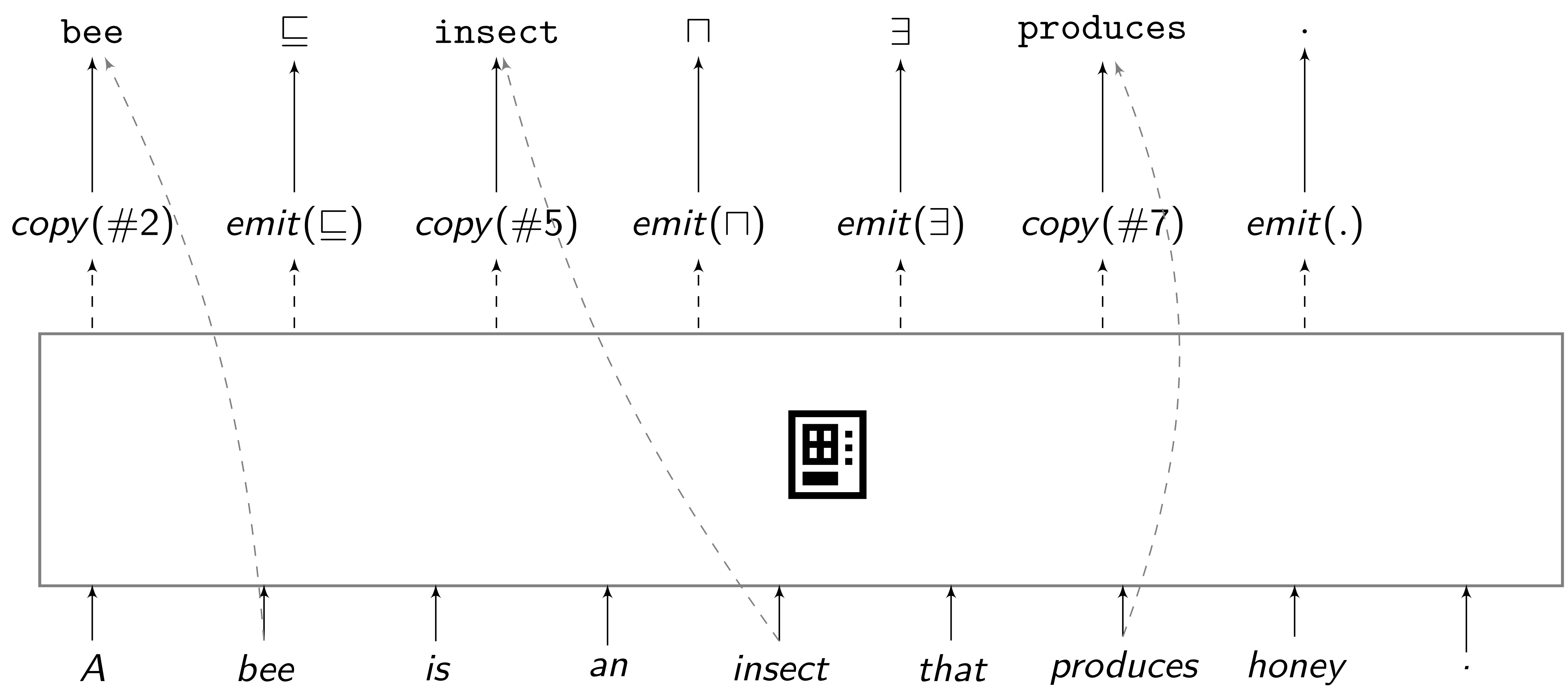


bee
⊑
insect
⊓
∃
copy(#2)
emit(⊑)
copy(#5)
emit(⊓)
emit(∃)
A
bee
is
an
insect
that
produces
honey
.

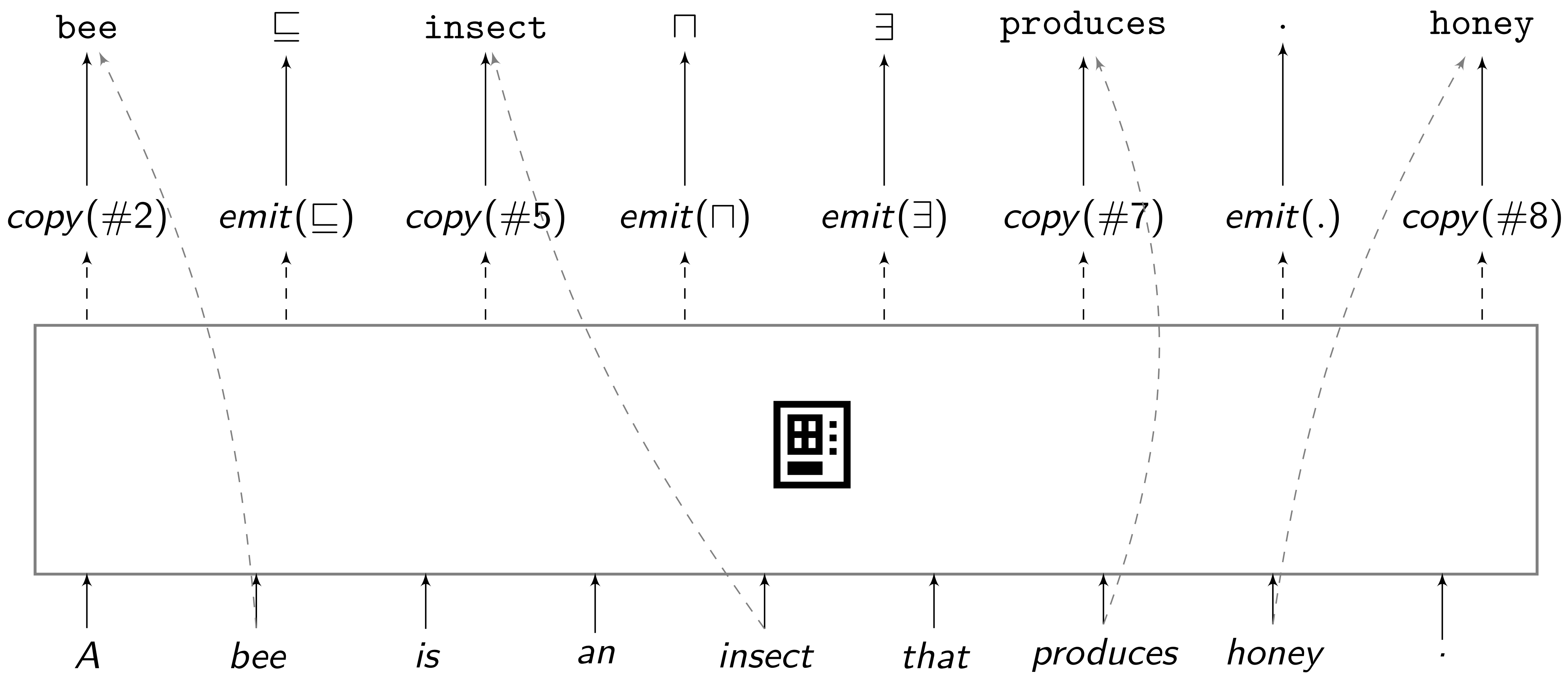
Translate


bee
⊑
insect
⊓
∃
produces
copy(#2)
emit(⊑)
copy(#5)
emit(⊓)
emit(∃)
copy(#7)
A
bee
is
an
insect
that
produces
honey
.

Translate

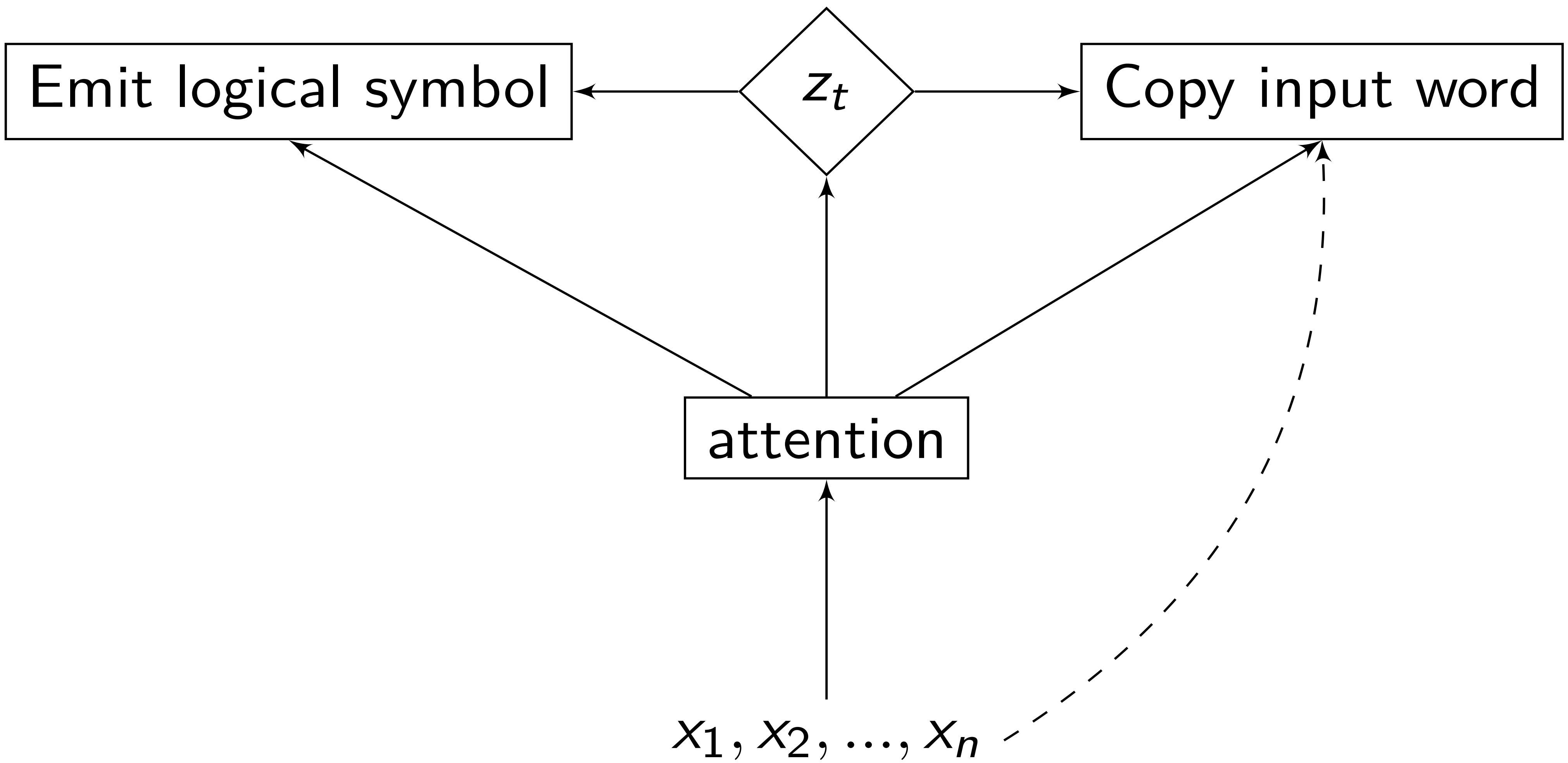

bee ⊑ insect ⊓ ∃ produces .
copy(#2) emit(⊑) copy(#5) emit(⊓) emit(∃) copy(#7) emit(.)
A bee is an insect that produces honey .

Translate



Quasi-zero vocabulary setting.

Translate



Emit logical symbol
z_t
Copy input word
attention
$x_1, x_2, \ldots, x_n$

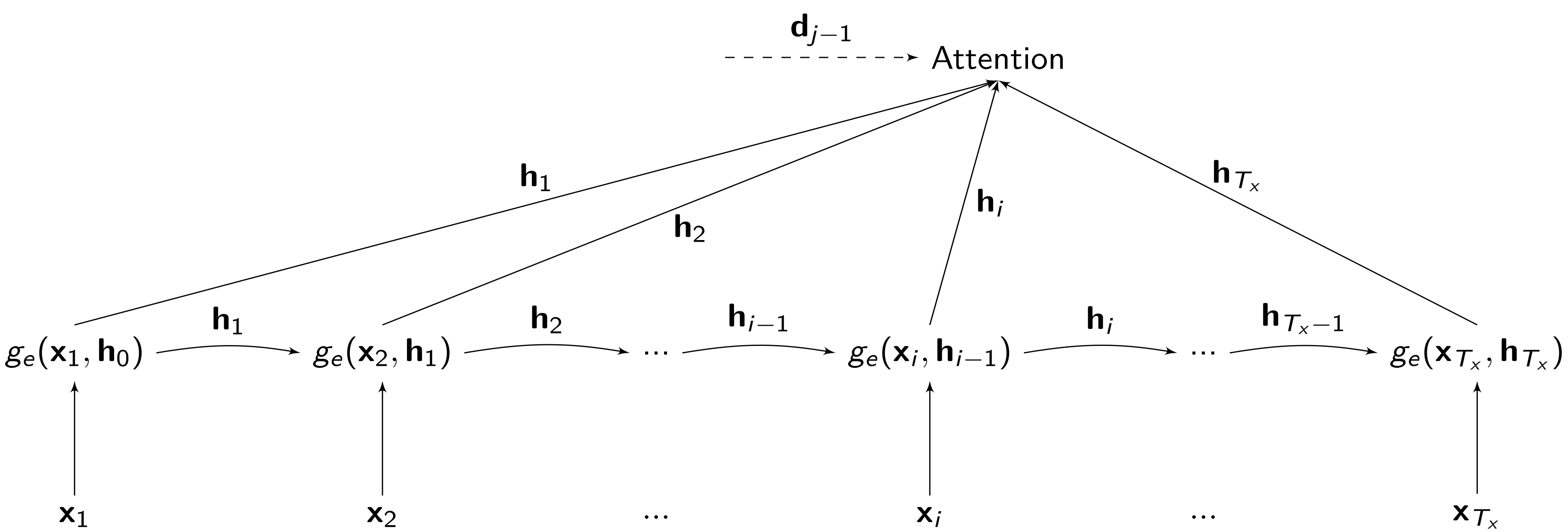
Translate

$\mathbf{x}_1$ $\mathbf{x}_2$ $\cdots$ $\mathbf{x}_i$ $\cdots$ $\mathbf{x}_{T_x}$

Translate

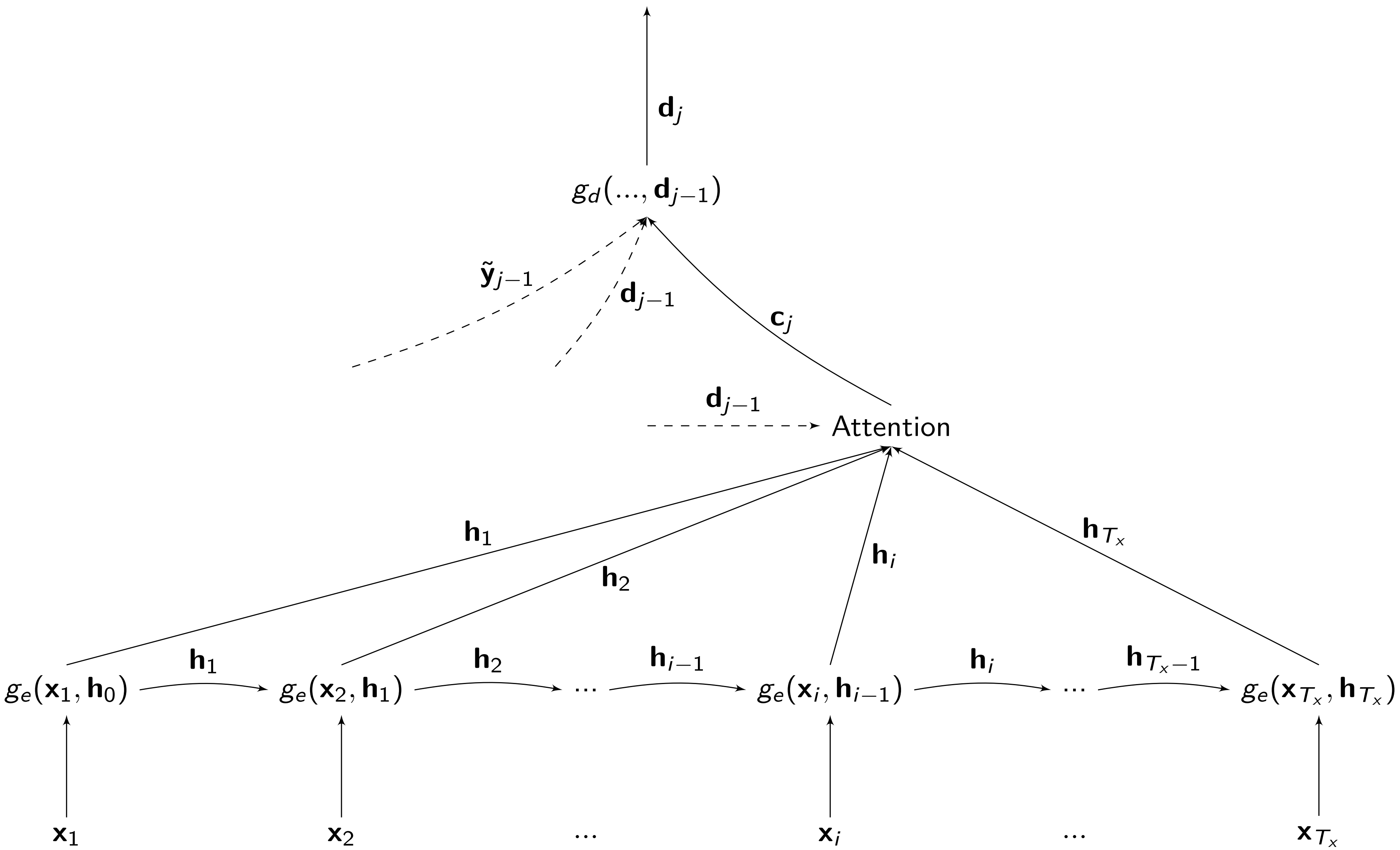
$$
\begin{tikzcd}
g_e(\mathbf{x}_1, \mathbf{h}_0) \arrow[r, bend left, "\mathbf{h}_1"] & g_e(\mathbf{x}_2, \mathbf{h}_1) \arrow[r, bend left, "\mathbf{h}_2"] & \cdots \arrow[r, bend left, "\mathbf{h}_{i-1}"] & g_e(\mathbf{x}_i, \mathbf{h}_{i-1}) \arrow[r, bend left, "\mathbf{h}_i"] & \cdots \arrow[r, bend left, "\mathbf{h}_{T_x-1}"] & g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x}) \\
\mathbf{x}_1 \arrow[u] & \mathbf{x}_2 \arrow[u] & \cdots & \mathbf{x}_i \arrow[u] & \cdots & \mathbf{x}_{T_x} \arrow[u]
\end{tikzcd}
$$

Translate



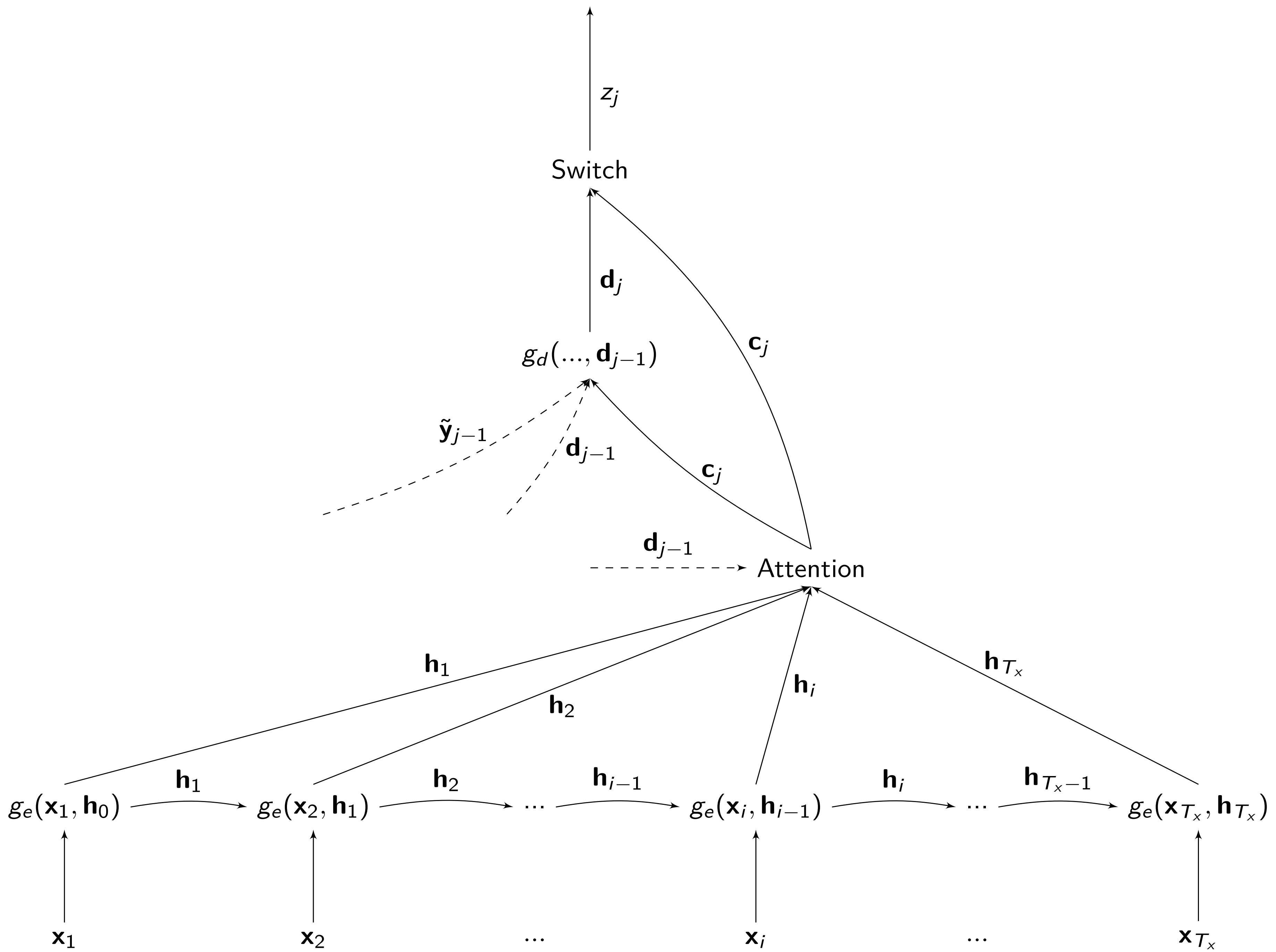
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_i$
$\mathbf{h}_{T_x}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$\mathbf{h}_1$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\mathbf{h}_2$
$\cdots$
$\mathbf{h}_{i-1}$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\mathbf{h}_i$
$\cdots$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\cdots$
$\mathbf{x}_i$
$\cdots$
$\mathbf{x}_{T_x}$

Translate



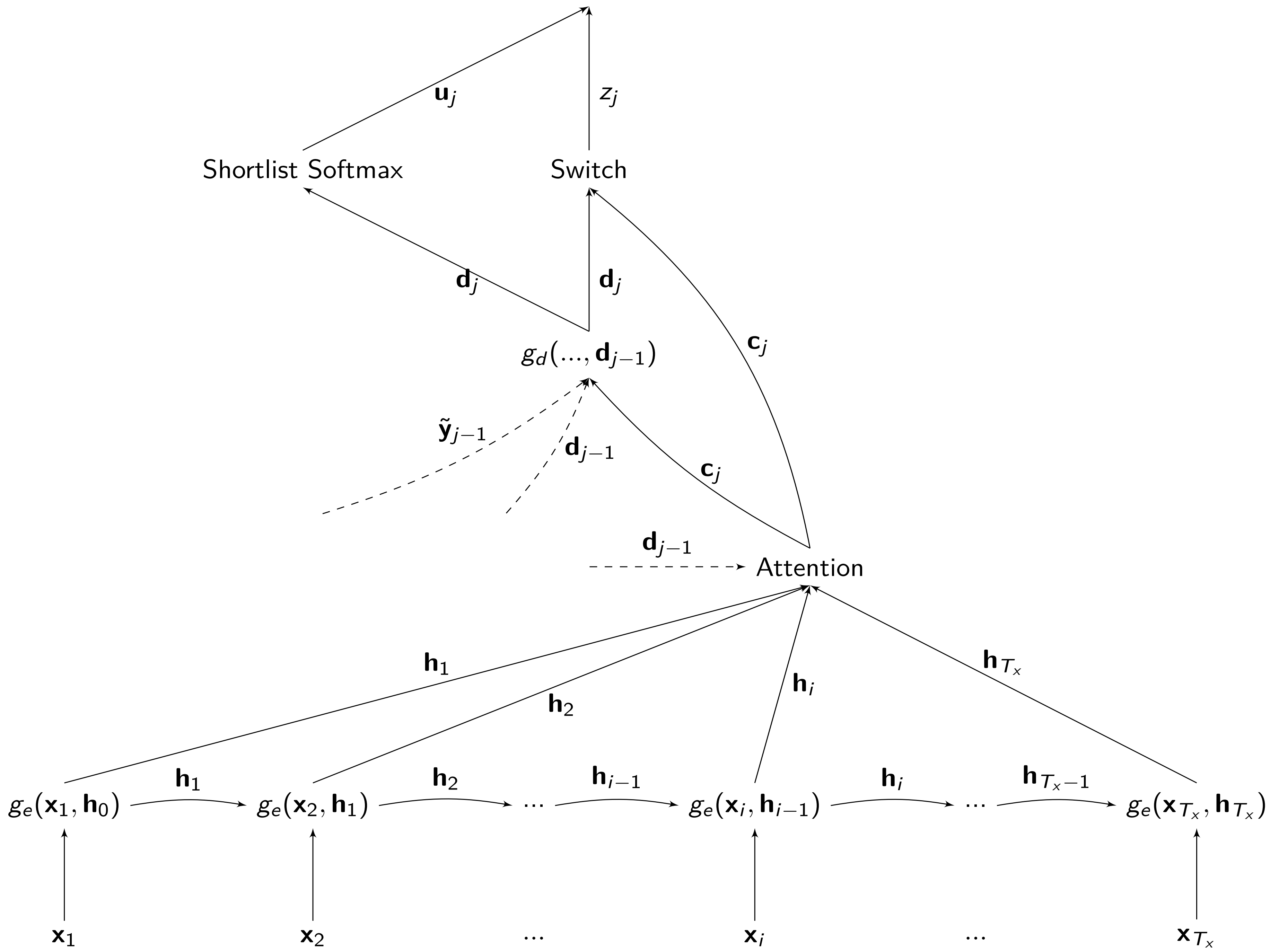
$\mathbf{d}_j$
$g_d(\ldots, \mathbf{d}_{j-1})$
$\tilde{\mathbf{y}}_{j-1}$
$\mathbf{d}_{j-1}$
$\mathbf{c}_j$
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_i$
$\mathbf{h}_{T_x}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$\mathbf{h}_1$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\mathbf{h}_2$
$\ldots$
$\mathbf{h}_{i-1}$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\mathbf{h}_i$
$\ldots$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\ldots$
$\mathbf{x}_i$
$\ldots$
$\mathbf{x}_{T_x}$

Translate



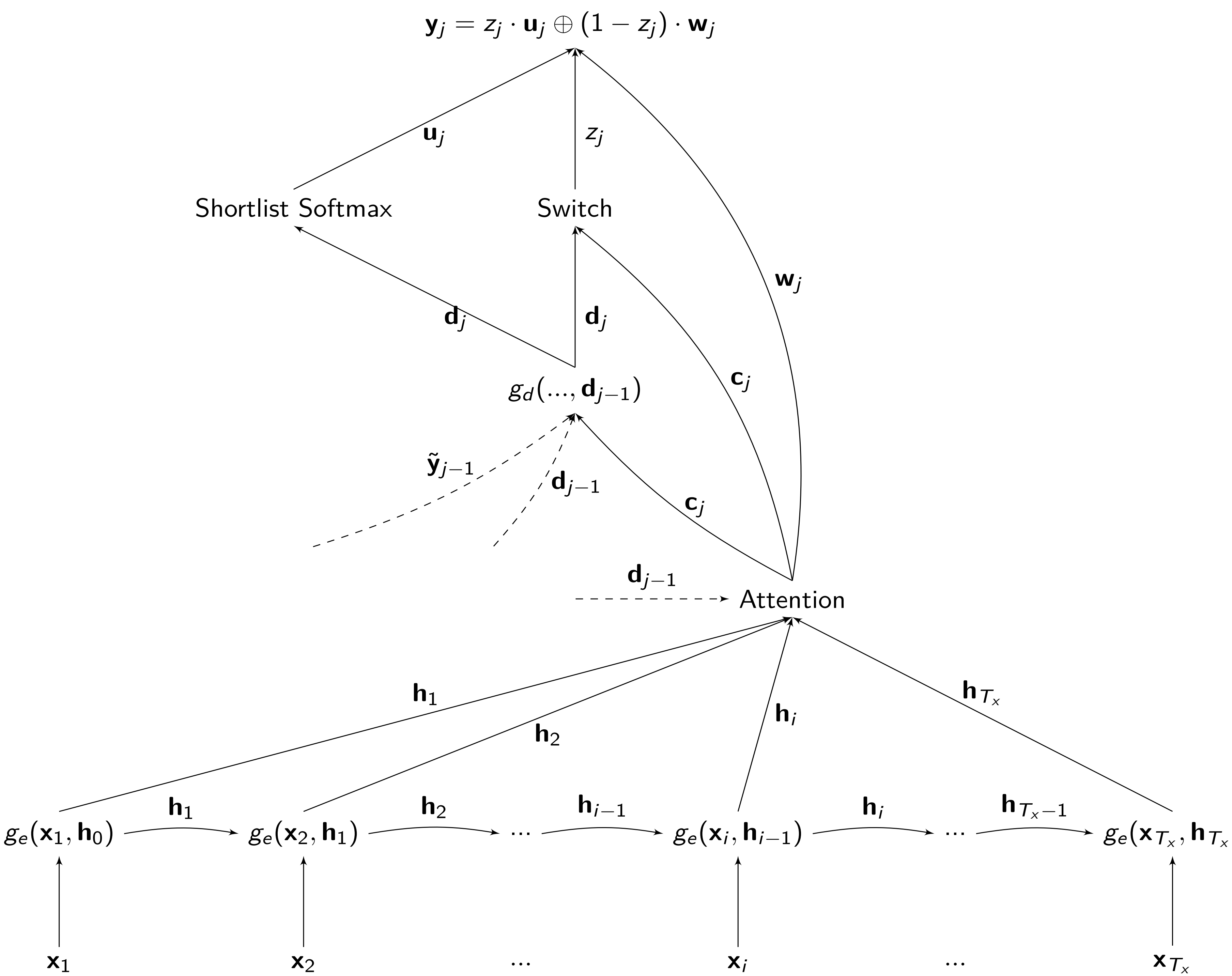
z_j
Switch
$\mathbf{d}_j$
$\mathbf{c}_j$
$g_d(\ldots, \mathbf{d}_{j-1})$
$\tilde{\mathbf{y}}_{j-1}$
$\mathbf{d}_{j-1}$
$\mathbf{c}_j$
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_{T_x}$
$\mathbf{h}_i$
$\mathbf{h}_2$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_{i-1}$
$\mathbf{h}_i$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\cdots$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\cdots$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\cdots$
$\mathbf{x}_i$
$\cdots$
$\mathbf{x}_{T_x}$

Translate

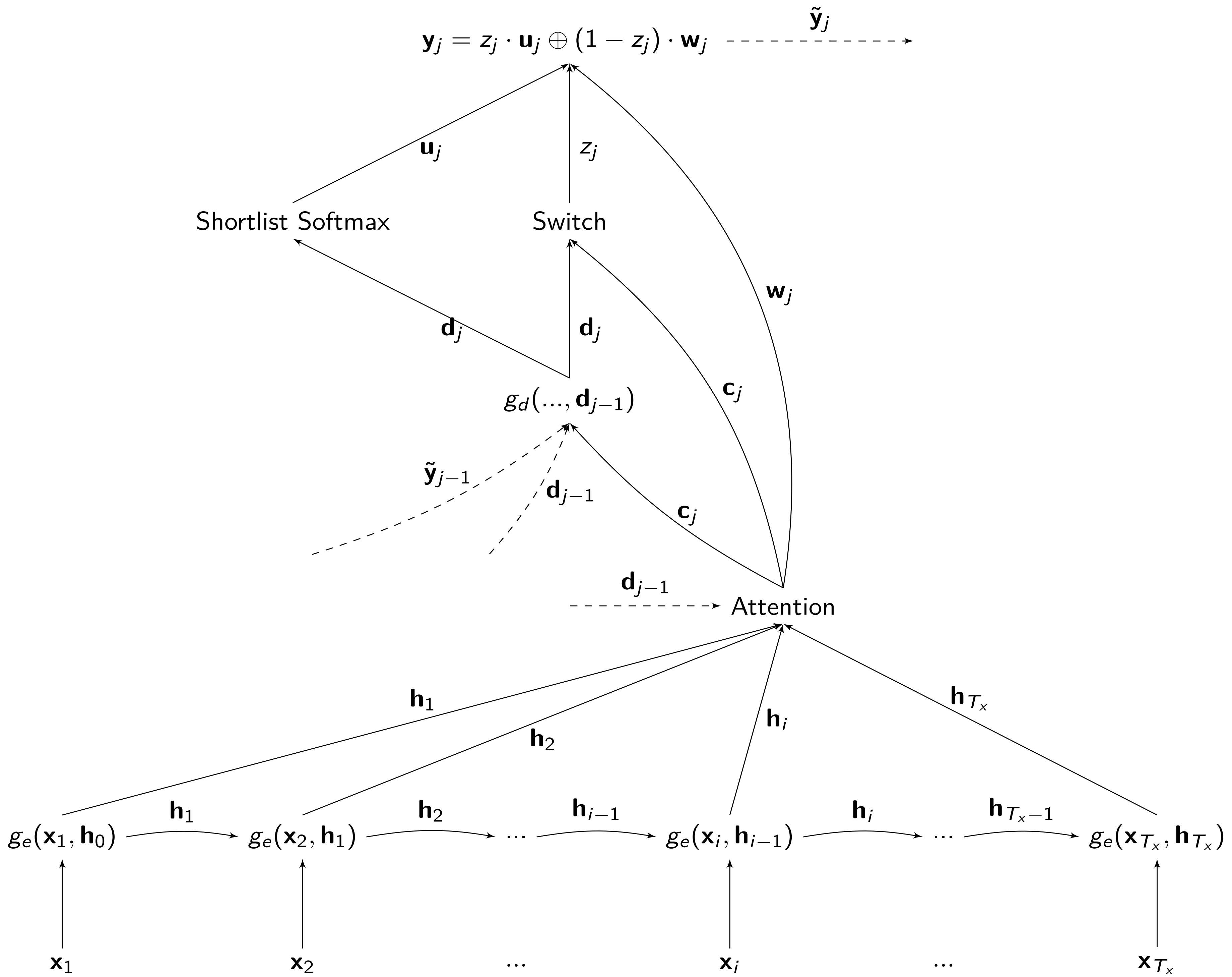


z_j
$\mathbf{u}_j$
Shortlist Softmax
Switch
$\mathbf{d}_j$
$\mathbf{d}_j$
$\mathbf{c}_j$
$g_d(\ldots, \mathbf{d}_{j-1})$
$\tilde{\mathbf{y}}_{j-1}$
$\mathbf{d}_{j-1}$
$\mathbf{c}_j$
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_{T_x}$
$\mathbf{h}_i$
$\mathbf{h}_2$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_{i-1}$
$\mathbf{h}_i$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\ldots$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\ldots$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\ldots$
$\mathbf{x}_i$
$\ldots$
$\mathbf{x}_{T_x}$

Translate


$\mathbf{y}_j = z_j \cdot \mathbf{u}_j \oplus (1 - z_j) \cdot \mathbf{w}_j$
$\mathbf{u}_j$
z_j
Shortlist Softmax
Switch
$\mathbf{w}_j$
$\mathbf{d}_j$
$\mathbf{d}_j$
$\mathbf{c}_j$
$g_d(\dots, \mathbf{d}_{j-1})$
$\tilde{\mathbf{y}}_{j-1}$
$\mathbf{d}_{j-1}$
$\mathbf{c}_j$
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_i$
$\mathbf{h}_{T_x}$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_{i-1}$
$\mathbf{h}_i$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\dots$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\dots$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\dots$
$\mathbf{x}_i$
$\dots$
$\mathbf{x}_{T_x}$

Translate



$\mathbf{y}_j = z_j \cdot \mathbf{u}_j \oplus (1 - z_j) \cdot \mathbf{w}_j$
$\tilde{\mathbf{y}}_j$
$\mathbf{u}_j$
z_j
Shortlist Softmax
Switch
$\mathbf{w}_j$
$\mathbf{d}_j$
$\mathbf{d}_j$
$\mathbf{c}_j$
$g_d(\ldots, \mathbf{d}_{j-1})$
$\tilde{\mathbf{y}}_{j-1}$
$\mathbf{d}_{j-1}$
$\mathbf{c}_j$
$\mathbf{d}_{j-1}$
Attention
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_i$
$\mathbf{h}_{T_x}$
$g_e(\mathbf{x}_1, \mathbf{h}_0)$
$\mathbf{h}_1$
$g_e(\mathbf{x}_2, \mathbf{h}_1)$
$\mathbf{h}_2$
$\ldots$
$\mathbf{h}_{i-1}$
$g_e(\mathbf{x}_i, \mathbf{h}_{i-1})$
$\mathbf{h}_i$
$\ldots$
$\mathbf{h}_{T_x-1}$
$g_e(\mathbf{x}_{T_x}, \mathbf{h}_{T_x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\ldots$
$\mathbf{x}_i$
$\ldots$
$\mathbf{x}_{T_x}$

Translate

RQ1. To what degree is the network capable to generalize over the syntactic structures of descriptive language? (many structures, one meaning)

RQ2. To what degree is the network capable to tolerate words that have not been seen during the training phase? (many meanings, one structure)

Translate: Closed-Vocabulary Evaluation

training set size	**FA**	**ED**	**TA**
2000	0.61	2.48	0.92
5000	0.84	0.60	0.98
10000	**0.89**	0.47	**0.99**
20000	0.81	**0.46**	0.98

Translate: Open-Vocabulary Evaluation

training set size	FA	ED	TA
2000	0.62	1.51	0.94
5000	0.86	0.63	0.98
10000	0.85	0.51	0.98
20000	**0.89**	**0.38**	**0.99**

Into the wild

So far, so good.

Into the wild

So far, so good. So what?

RQ3. To what extent is the model capable to improve its performances with the addition of few annotated examples?

The Reference Set

500 manually curated examples from well known ontologies or formalized *ad hoc* by knowledge engineers.

The Reference Set

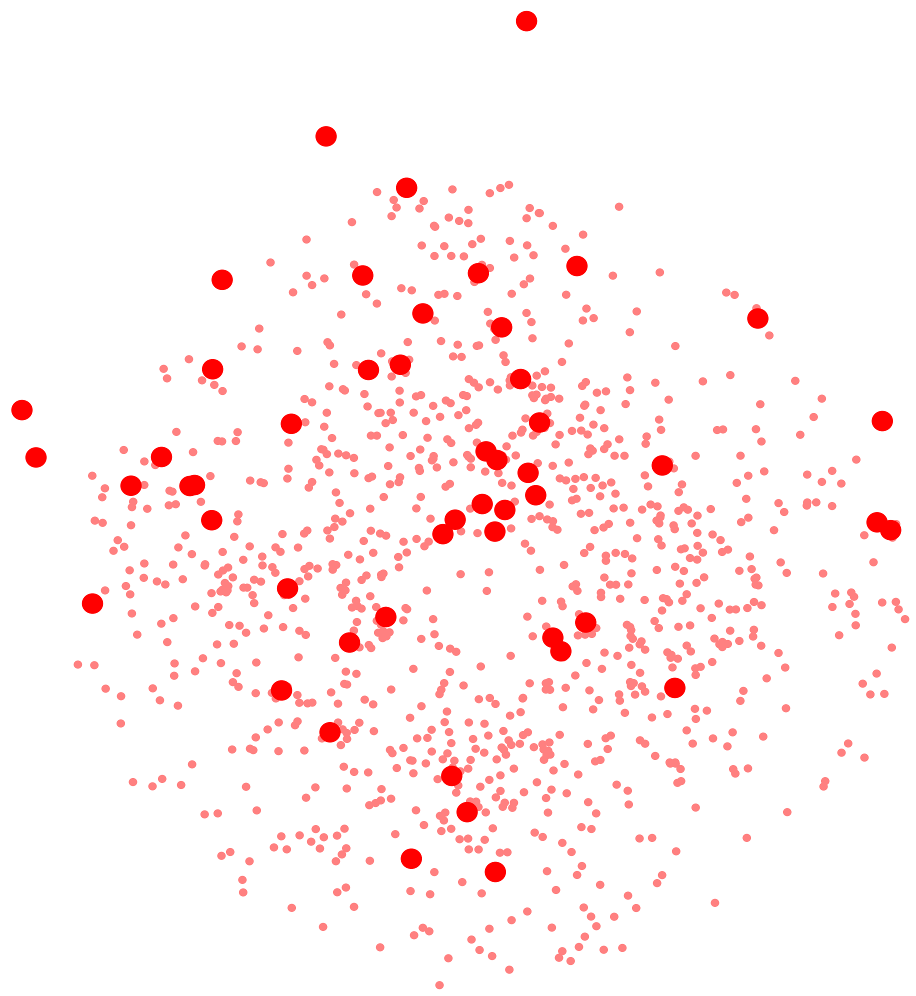
500 manually curated examples from well known ontologies or formalized *ad hoc* by knowledge engineers.

	size	**len.**	**LEN.**	**avg. len.**	**exist.**	**univ.**	**card. restr.**
training	75	5	28	11.72	42.67%	2.67%	9.33%
test	425	5	40	12.36	50.82%	4.47%	9.18%

Evaluation Against the Reference Set

system	CF	FA	ED	TA
Grammar Parser	17	0.04	-	-
Tag&Transduce	0	0.00	11.7	0.10
Translate (20k-open)	**38**	0.09	4.55	0.49

Evaluation Against the Reference Set



Evaluation Against the Reference Set

training set size	CF	FA	ED	TA
2k	35	0.08	4.80	0.47
2k+75	143	0.34	3.44	0.60
5k	38	0.09	4.58	0.48
5k+75	126	0.30	3.55	0.59
10k	39	0.09	4.59	0.48
10k+75	82	0.19	4.06	0.55
20k	38	0.09	4.55	0.49
20k+75	55	0.13	4.53	0.50

Through the Looking Glass

Contributions:

- suitable architecture;
- bootstrap datasets and reference set;
- a new approach.

Lesson Learned.

- the pointing network is a powerful architecture and can deal successfully with our quasi-zero vocabulary setting;
- the bootstrap data can be a good start, but the model can be biased in the perspective of an adaptation to real world data;
- the model could learn from raw text (with a minimum preprocessing), though, on the long term it would require a large amount of text.

The Road Ahead

Future work:

- more on the architecture: Bi-GRU, LSTMs, ...;
- more on the data: definition extraction, distant supervision, generative autoencoding, ...;
- less on the radical end-to-end and zero feature engineering.

UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Learning to Learn Concept Descriptions

Giulio Petrucci

http://bit.ly/giuliophd

Advisor
Dr. **Marco Rospocher**
Fondazione Bruno Kessler-IRST

Co-Advisor
Dr. **Chiara Ghidini**
Fondazione Bruno Kessler-IRST

Committee
Prof. Dr. **Philipp Cimiano**
Bielefeld University
Prof. Dr. **Simone Paolo Ponzetto**
Universität Mannheim
Prof. Dr. **Roberto Basili**
Università degli Studi di Roma
"Tor Vergata"

January 2018

JPARK
pikes.fbk.eu/jpark

PSL4EA
pikes.fbk.eu/psl4ea

KE4IR
pikes.fbk.eu/ke4ir

PIKES
pikes.fbk.eu

KnowledgeStore
knowledgestore.fbk.eu

RDFpro
rdfpro.fbk.eu

Marco Rospocher

FONDAZIONE BRUNO KESSLER

rospocher@fbk.eu
dkm.fbk.eu/rospocher
@marcorospocher

BPMN Ontology
dkm.fbk.eu/bpmn-ontology

PESCaDO
bit.ly/pescado-onto

NewsReader
Event & Situation Ontology
github.com/newsreader/eso

MoKi
the Modelling WiKi ---
moki.fbk.eu

TEX-OWL
github.com/dkmfbk/TexOwl

PREMON
premon.fbk.eu