Towards Probabilistic Acceptors and Transducers for Feature Structures

Daniel Quernheim

Institute for Natural Language Processing, University of Stuttgart

daniel@ims.uni-stuttgart.de

Kevin Knight

Information Sciences Institute, University of Southern California

knight@isi.edu

Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6)

July 12, 2012

Linguistic structures

Linguistic structures

Strings

surface forms, phonology, morphology

Linguistic structures

Strings

surface forms, phonology, morphology

Trees

syntax

Linguistic structures

Strings

surface forms, phonology, morphology

Trees

syntax

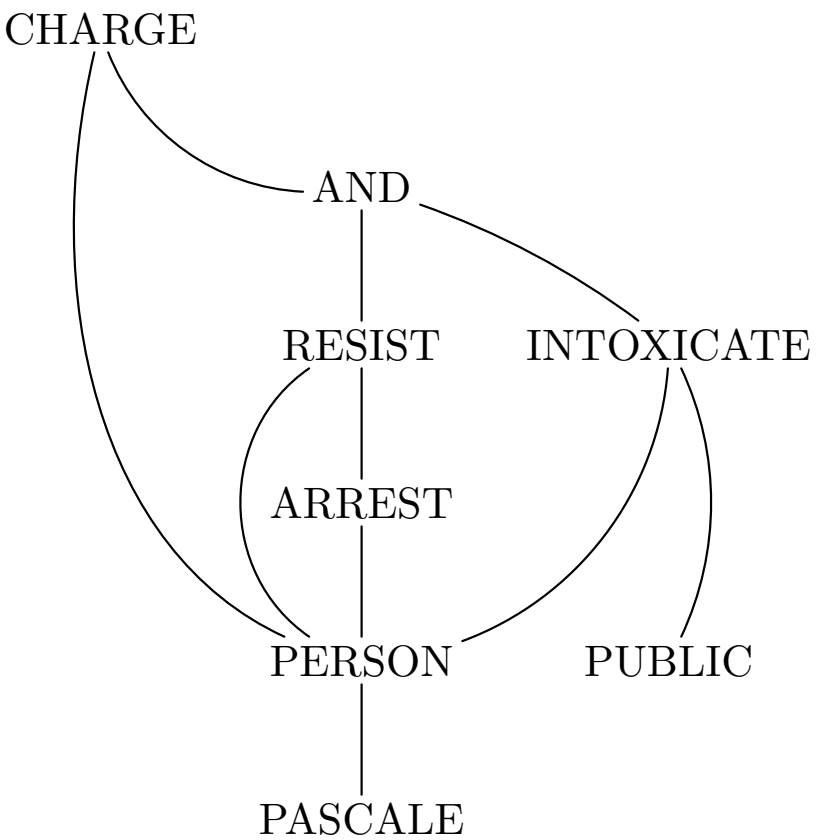
Feature structures (= directed acyclic graphs)

deep syntax (LFG etc.)
semantics (abstract meaning representations)

Feature structures

$$
\begin{bmatrix}
\text{INSTANCE} & \text{charge} \\
\text{THEME} & \boxed{1}\begin{bmatrix}\text{INSTANCE} & \text{person} \\ \text{NAME} & \text{“Pascale”}\end{bmatrix} \\
\text{PRED} & \begin{bmatrix}
\text{INSTANCE} & \text{and} \\
\text{OP1} & \begin{bmatrix}\text{INSTANCE} & \text{resist} \\ \text{AGENT} & \boxed{1} \\ \text{THEME} & \begin{bmatrix}\text{INSTANCE} & \text{arrest} \\ \text{THEME} & \boxed{1}\end{bmatrix}\end{bmatrix} \\
\text{OP2} & \begin{bmatrix}\text{INSTANCE} & \text{intoxicate} \\ \text{THEME} & \boxed{1} \\ \text{LOCATION} & \begin{bmatrix}\text{INSTANCE} & \text{public}\end{bmatrix}\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Directed acyclic graphs



$$\begin{aligned}
\text{CHARGE} &\mapsto \text{charge}(\text{theme}, \text{pred}) \\
\text{AND} &\mapsto \text{and}(\text{op1}, \text{op2}) \\
\text{RESIST} &\mapsto \text{resist}(\text{agent}, \text{theme}) \\
\text{ARREST} &\mapsto \text{arrest}(\text{theme}) \\
\text{INTOXICATE} &\mapsto \text{intoxicate} \\
&\qquad (\text{theme}, \text{location}) \\
\text{PUBLIC} &\mapsto \text{public}() \\
\text{PERSON} &\mapsto \text{person}(\text{name}) \\
\text{PASCALE} &\mapsto \text{"Pascale"}
\end{aligned}$$

Translation pipelines

Syntax-based MT pipeline

$$\text{fstring} \rightarrow \boxed{\text{translate}} \rightarrow \text{etree} \rightarrow \boxed{\text{language model}} \rightarrow \text{estring}$$

Translation pipelines

Syntax-based MT pipeline

$$\text{fstring} \rightarrow \boxed{\text{translate}} \rightarrow \text{etree} \rightarrow \boxed{\text{language model}} \rightarrow \text{estring}$$

- The individual components are efficiently represented as **weighted tree acceptors and transducers**.

Translation pipelines

Syntax-based MT pipeline

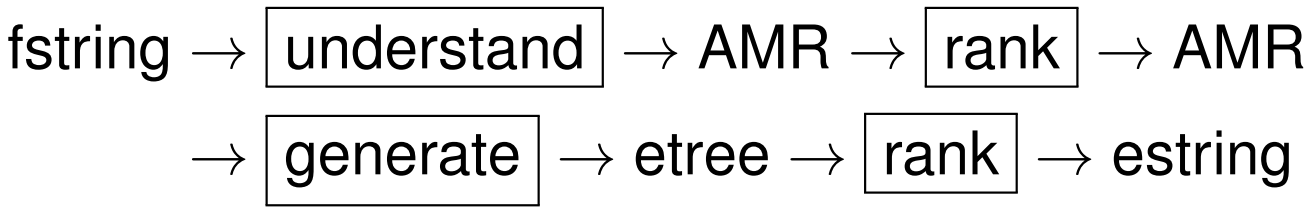
$$\underset{\text{fstring}}{\text{FSA}} \rightarrow \boxed{\underset{\text{translate}}{\text{nl-XTOPs}^{-1}}} \rightarrow \underset{\text{etree}}{\text{RTG}} \rightarrow \boxed{\underset{\text{language model}}{\text{FSA}}} \rightarrow \underset{\text{estring}}{\text{FSA}}$$

- The individual components are efficiently represented as **weighted tree acceptors and transducers**.

$$\text{estring} = \textsc{BestPath}(\textsc{Intersect}(\text{language model}, \textsc{Yield}(\textsc{Backwards}(\text{translate}, \text{fstring})))).$$

Translation pipelines (2)

Semantics-based MT pipeline



Translation pipelines (2)

Semantics-based MT pipeline

$$\text{fstring} \rightarrow \boxed{\text{understand}} \rightarrow \text{AMR} \rightarrow \boxed{\text{rank}} \rightarrow \text{AMR}$$
$$\rightarrow \boxed{\text{generate}} \rightarrow \text{etree} \rightarrow \boxed{\text{rank}} \rightarrow \text{estring}$$

- No suitable automaton framework is known!

Algorithms and automata

	string automata	tree automata	graph automata
k-best	paths through a WFSA	trees in a weighted forest	
EM training	Forward-backward EM	Tree transducer EM training	
Determinization	of weighted string acceptors	of weighted tree acceptors	
Transducer composition	WFST composition	Many transducers not closed under composition	
General tools	AT&T FSM, Carmel, OpenFST	Tiburon	

Table: General-purpose algorithms for strings, trees and feature structures.

Algorithms and automata

	string automata	tree automata	graph automata
k-best	paths through a WFSA	trees in a weighted forest	?
EM training	Forward-backward EM	Tree transducer EM training	?
Determinization	of weighted string acceptors	of weighted tree acceptors	?
Transducer composition	WFST composition	Many transducers not closed under composition	?
General tools	AT&T FSM, Carmel, OpenFST	Tiburon	?

Table: General-purpose algorithms for strings, trees and feature structures.

Algorithms and automata (2)

Our goal

- Find an adequate automaton model for the pipeline parts
- Investigate algorithms and fill all the blanks!

Algorithms and automata (2)

Our goal

- Find an adequate automaton model for the pipeline parts
- Investigate algorithms and fill all the blanks!

Candidates

- Treating everything as a tree (too weak?)
- Unification grammars (HPSG, LFG) (too powerful?)
- Hyperedge replacement grammar (too powerful?)

Algorithms and automata (2)

Our goal

- Find an adequate automaton model for the pipeline parts
- Investigate algorithms and fill all the blanks!

Candidates

- Treating everything as a tree (too weak?)
- Unification grammars (HPSG, LFG) (too powerful?)
- Hyperedge replacement grammar (too powerful?)
- Some straightforward extension of string/tree automata?

Dag automata

finite string automaton: (FSA)

one input state, one input symbol, one output state

$$\cdots \quad p \text{——} \sigma \text{——} q \quad \cdots$$

Dag automata

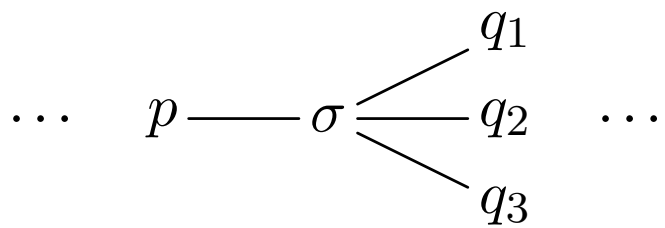
finite string automaton: (FSA)

one input state, one input symbol, one output state

$$\cdots \quad p \text{ —— } \sigma \text{ —— } q \quad \cdots$$

finite tree automaton: (FTA)

one input state, one input symbol, many output states



Dag automata

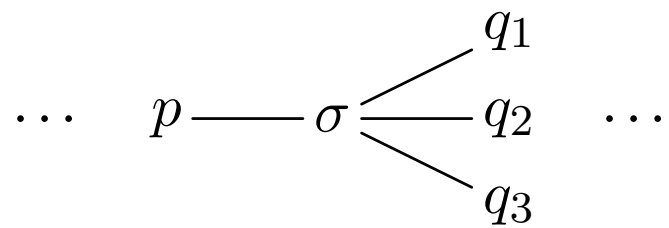
finite string automaton: (FSA)

one input state, one input symbol, one output state

$$\cdots \quad p \text{ —— } \sigma \text{ —— } q \quad \cdots$$

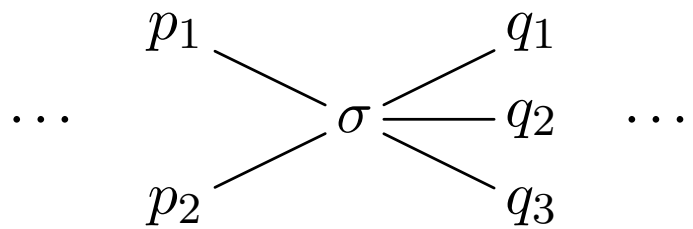
finite tree automaton: (FTA)

one input state, one input symbol, many output states



finite dag automaton: (FDA?)

many input states, one input symbol, many output states



Dag automata (2)

KAMIMURA and SLUTZKI (1981, 1982)

- Dag acceptors and dag-to-tree transducers
- They proved a couple of technical properties, no algorithms

Dag automata (2)

KAMIMURA and SLUTZKI (1981, 1982)

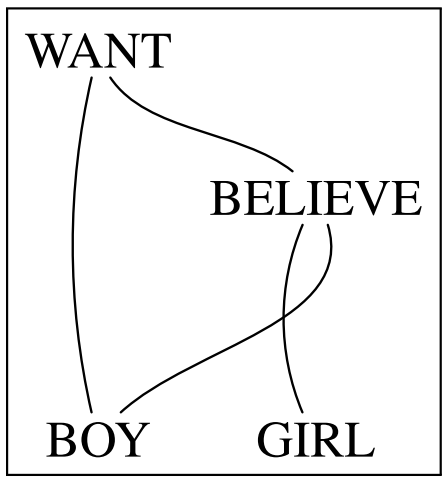
- Dag acceptors and dag-to-tree transducers
- They proved a couple of technical properties, no algorithms
- We investigate their model with some adjustments:

  - not only adjacent leaves can be connected

Dag automata (2)

KAMIMURA and SLUTZKI (1981, 1982)

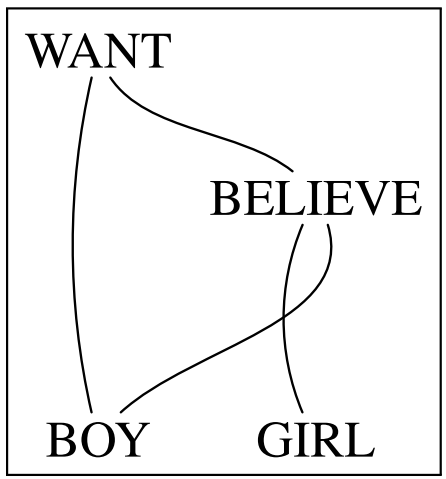
- Dag acceptors and dag-to-tree transducers
- They proved a couple of technical properties, no algorithms
- We investigate their model with some adjustments:
  - not only adjacent leaves can be connected



Dag automata (2)

KAMIMURA and SLUTZKI (1981, 1982)

- Dag acceptors and dag-to-tree transducers
- They proved a couple of technical properties, no algorithms
- We investigate their model with some adjustments:
  - not only adjacent leaves can be connected
  - top-down transducers instead of bottom-up
  - we introduce weights (probabilities)



Example dag automaton

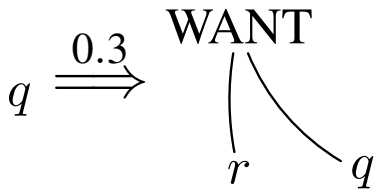
$$
\begin{aligned}
q &\rightarrow \textsc{want}(r, q)\langle 0.3\rangle \\
q &\rightarrow \textsc{believe}(r, q)\langle 0.2\rangle \\
q &\rightarrow r\,\langle 0.4\rangle \mid \emptyset\,\langle 0.1\rangle \\
r &\rightarrow \textsc{boy}\langle 0.3\rangle \mid \textsc{girl} \\
&\quad\ \langle 0.3\rangle \mid \emptyset\,\langle 0.1\rangle \\
[r, r] &\rightarrow r\,\langle 0.2\rangle \\
[r, r, r] &\rightarrow r\,\langle 0.1\rangle
\end{aligned}
$$

$$
\begin{aligned}
\textsc{want} &\mapsto \mathsf{want}(\mathsf{agent}, \mathsf{theme}) \\
\textsc{believe} &\mapsto \mathsf{believe}(\mathsf{agent}, \mathsf{theme}) \\
\textsc{boy} &\mapsto \mathsf{boy}() \\
\textsc{girl} &\mapsto \mathsf{girl}()
\end{aligned}
$$

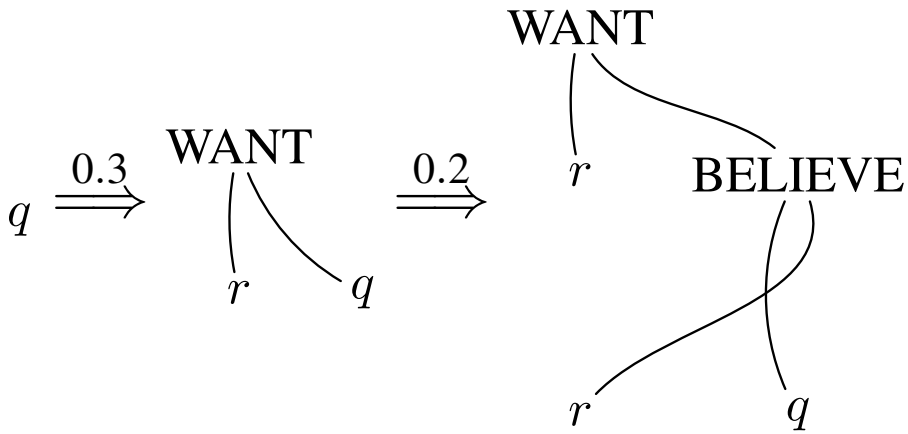
Example dag generation

q

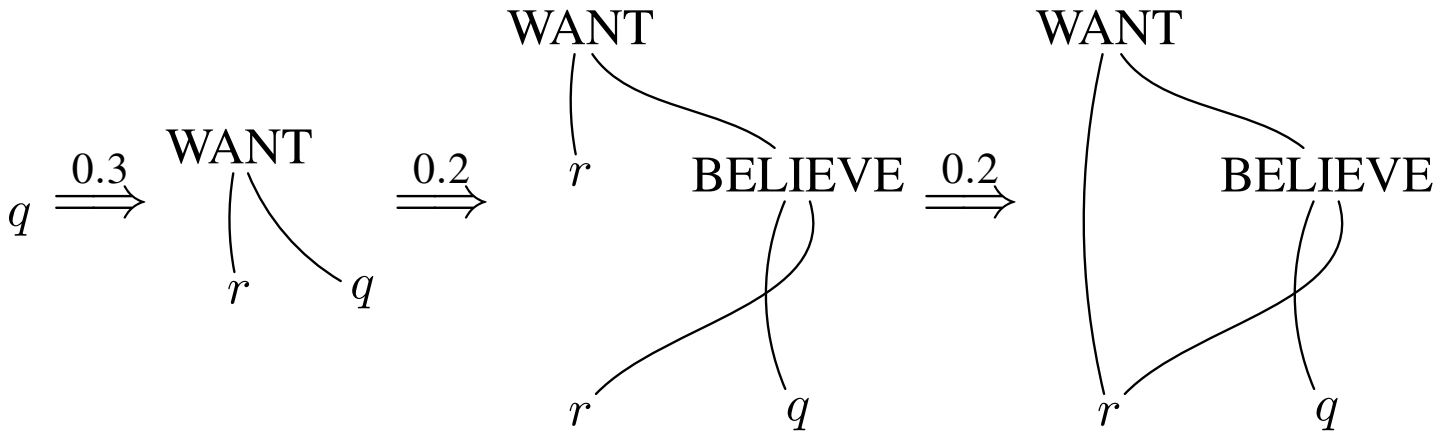
Example dag generation



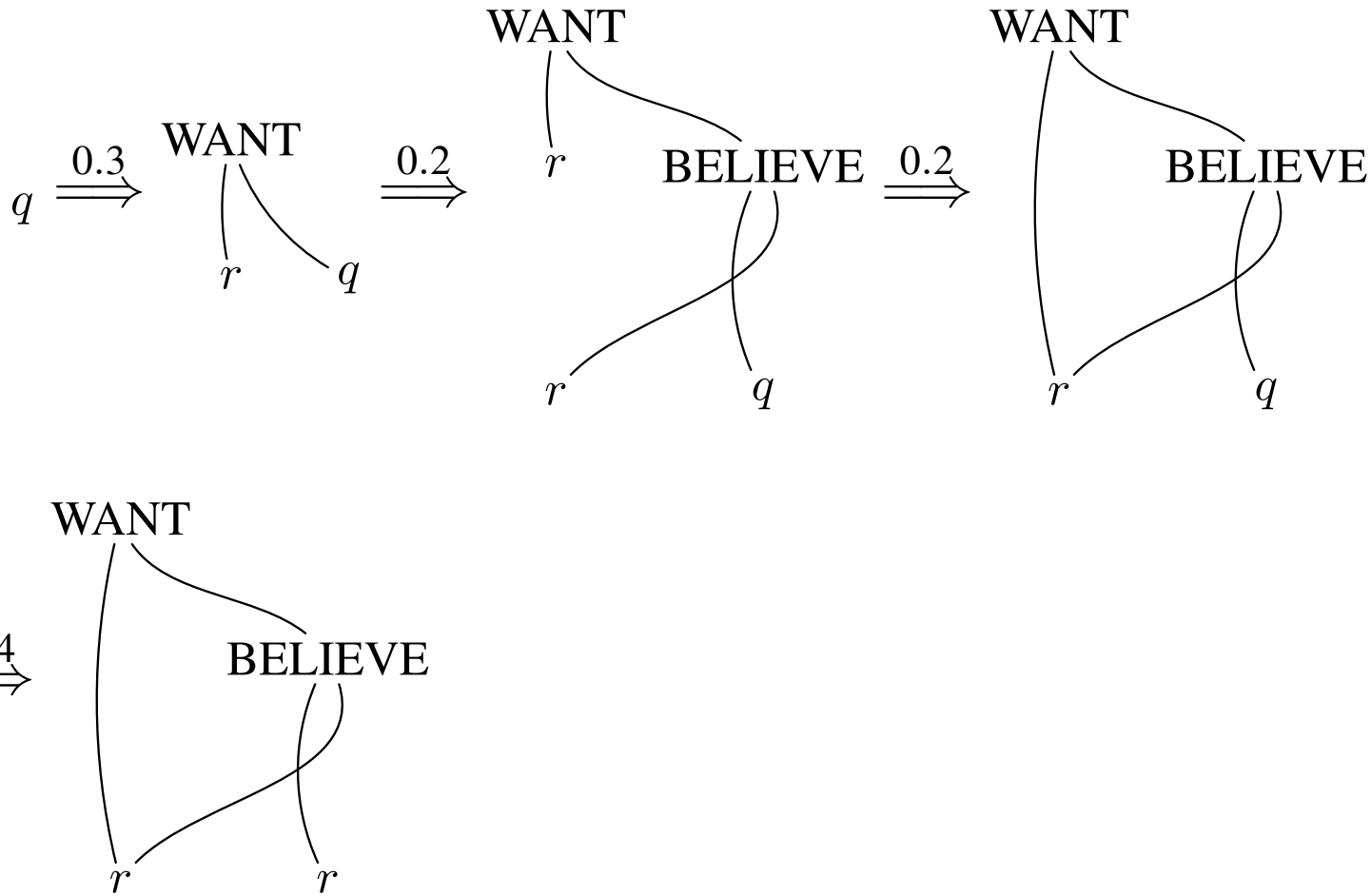
Example dag generation



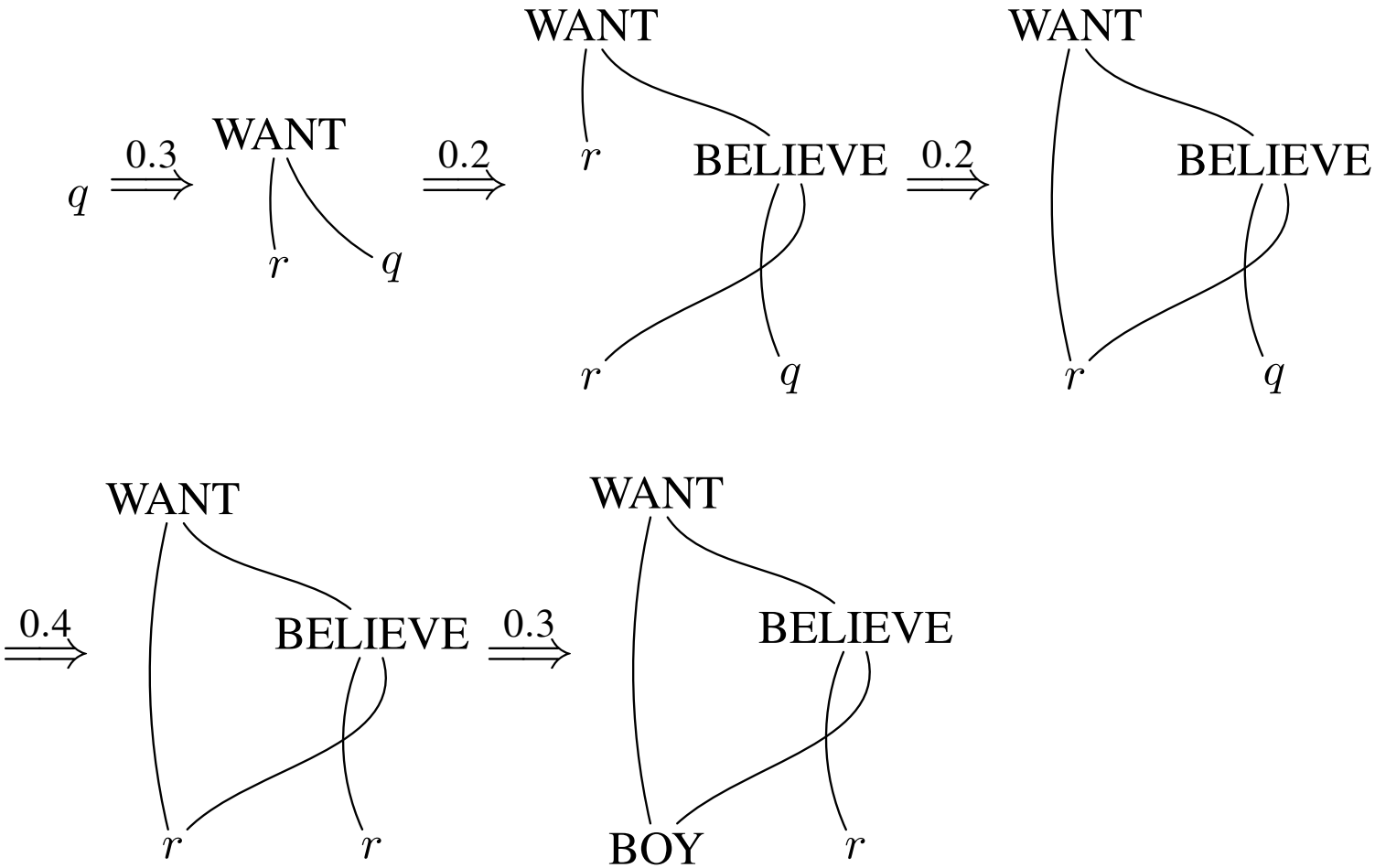
Example dag generation



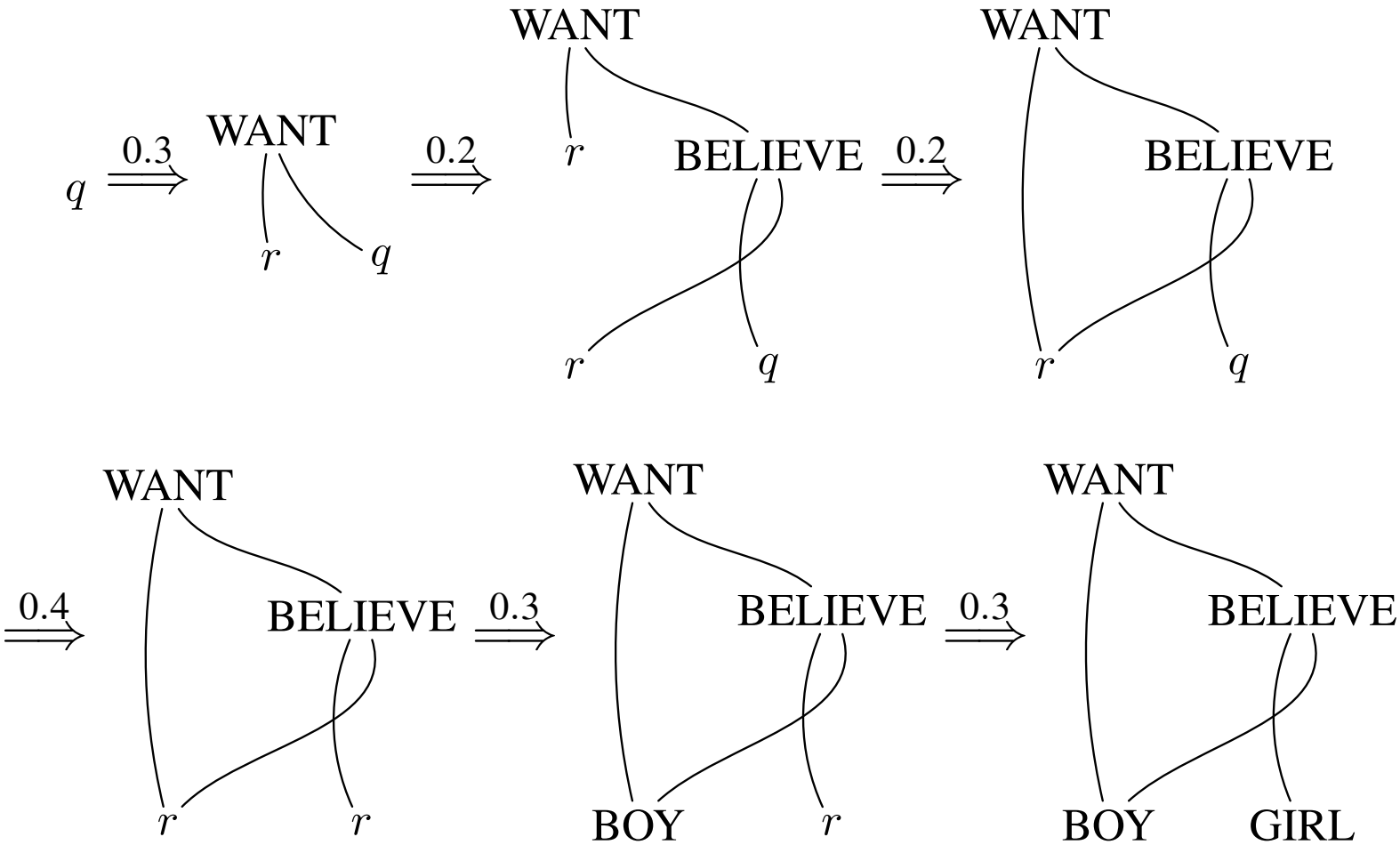
Example dag generation



Example dag generation



Example dag generation



Example dag transducer rules

- Rules have m incoming edges with states and produce m trees

Example dag transducer rules

- Rules have m incoming edges with states and produce m trees

$$[q_{nomb}, q_{accb}].\text{BOY} \rightarrow \text{NP(the boy)}, \text{NP(him)}$$
$$q_{accg}.\text{GIRL} \rightarrow \text{NP(the girl)}$$

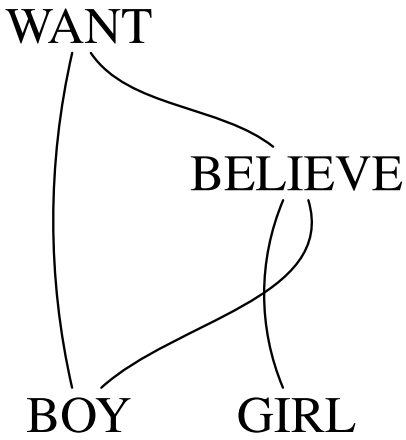
Example dag transducer rules

- Rules have m incoming edges with states and produce m trees
- Rules have n outgoing edges and n variables to pass states down

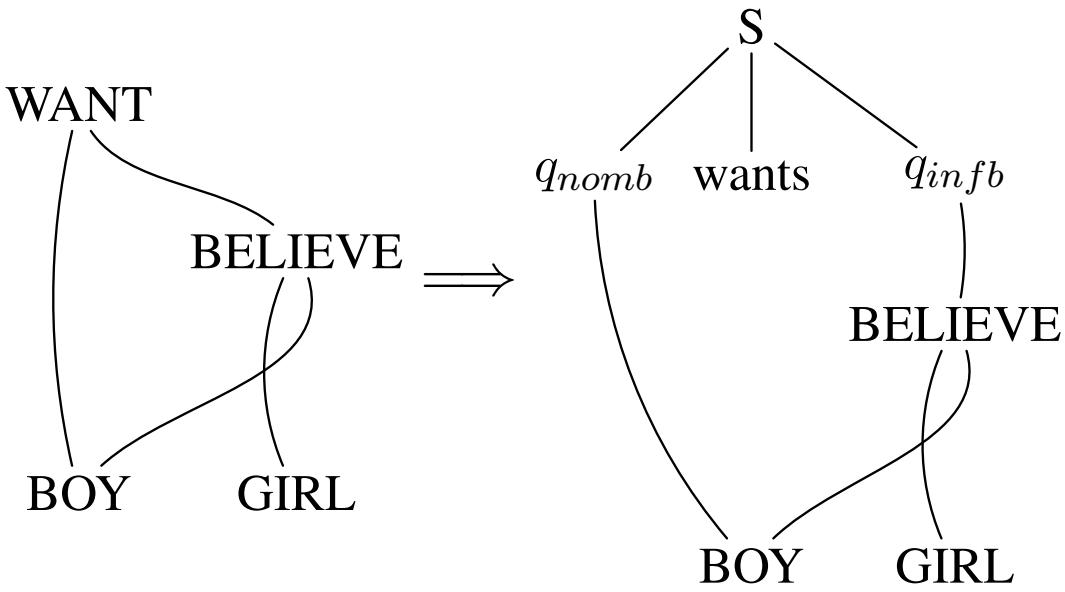
$$[q_{nomb}, q_{accb}].\text{BOY} \rightarrow \text{NP(the boy), NP(him)}$$
$$q_{accg}.\text{GIRL} \rightarrow \text{NP(the girl)}$$

$$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{wants}, q_{infb}.y)$$
$$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to believe}, q_{accb}.y)$$

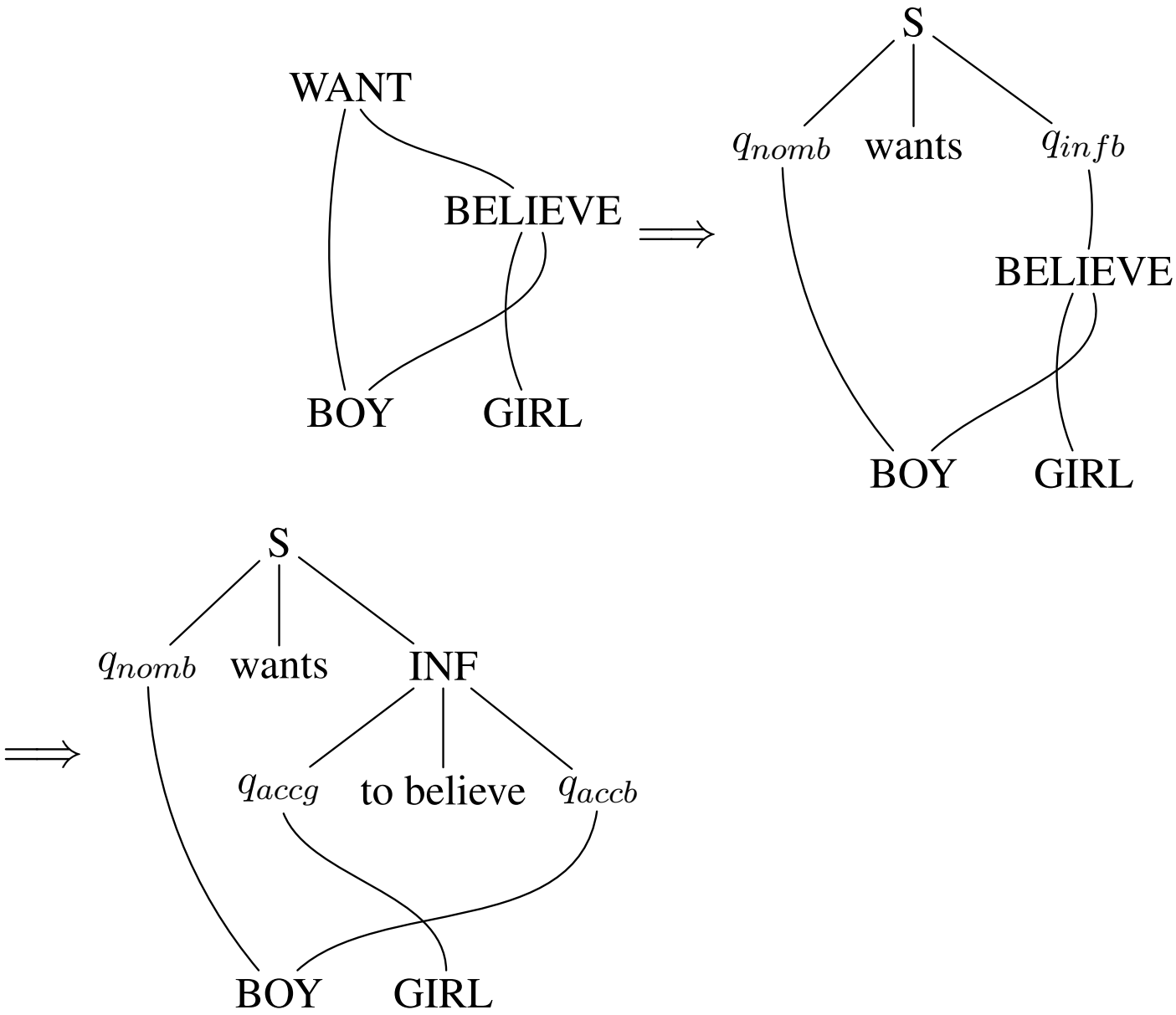
Example dag transduction



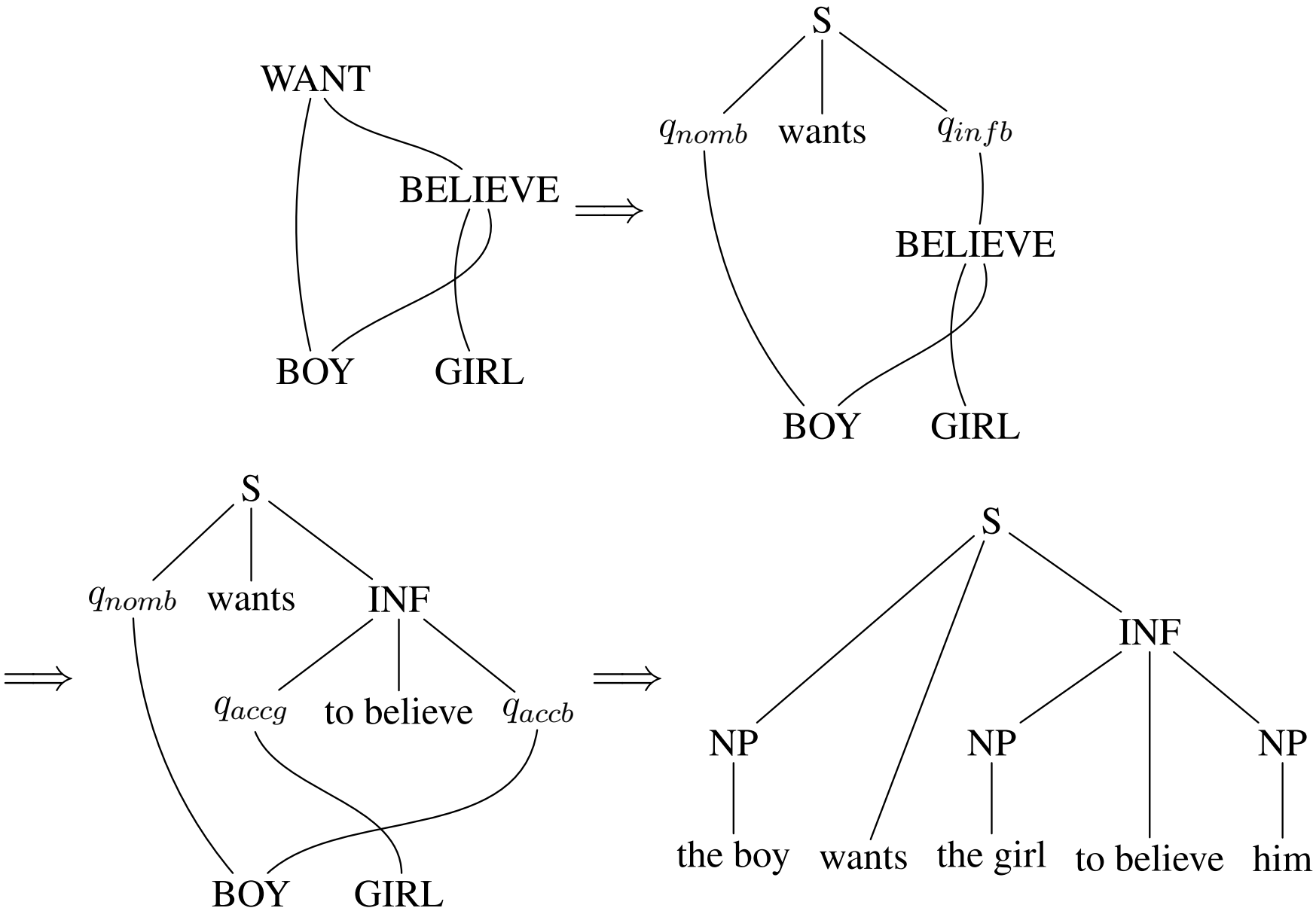
Example dag transduction



Example dag transduction



Example dag transduction



What's it for?

What's wrong with phrase-based and syntax-based MT?

- We want the "**who did what to whom, when, where, and why**"
- Preservation of meaning can be more important than grammaticality/fluency
- We are aiming for **useful** translation!

What's it for?

What's wrong with phrase-based and syntax-based MT?

- We want the "**who did what to whom, when, where, and why**"
- Preservation of meaning can be more important than grammaticality/fluency
- We are aiming for **useful** translation!

But haven't people tried and failed?

Yes, but...

What's it for?

What's wrong with phrase-based and syntax-based MT?

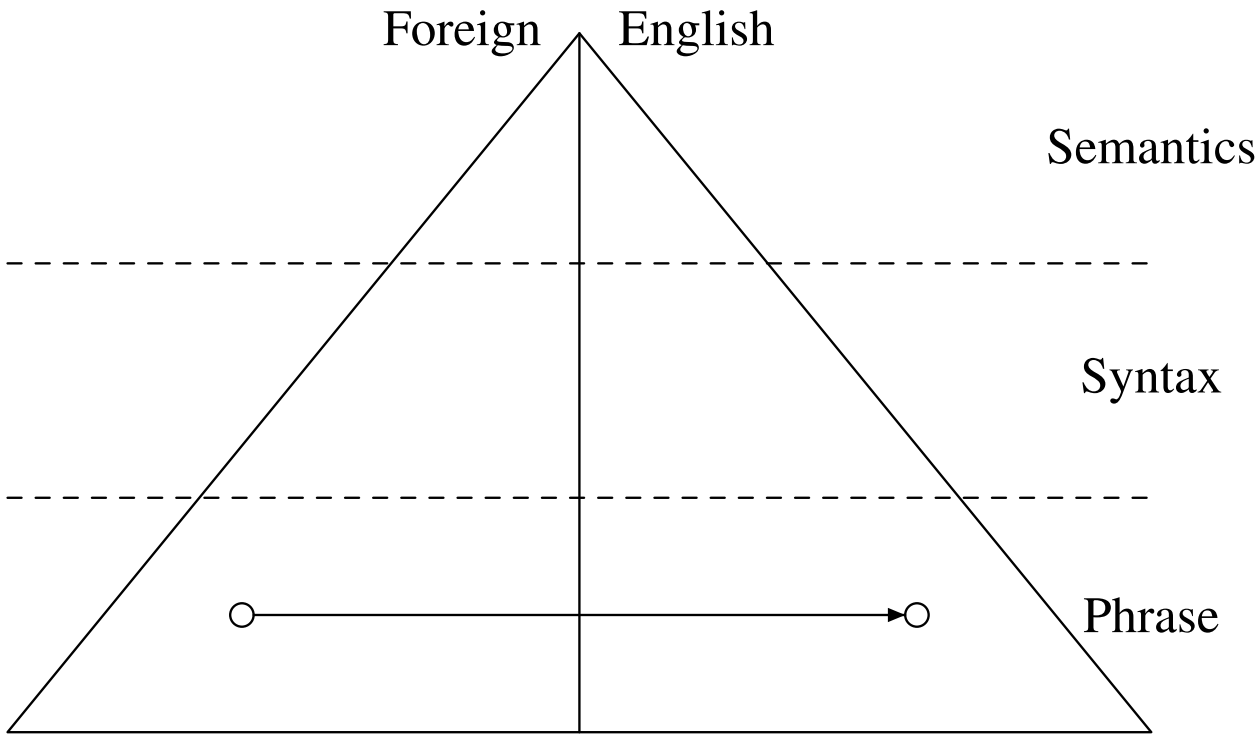
- We want the "**who did what to whom, when, where, and why**"
- Preservation of meaning can be more important than grammaticality/fluency
- We are aiming for **useful** translation!

But haven't people tried and failed?

Yes, but...

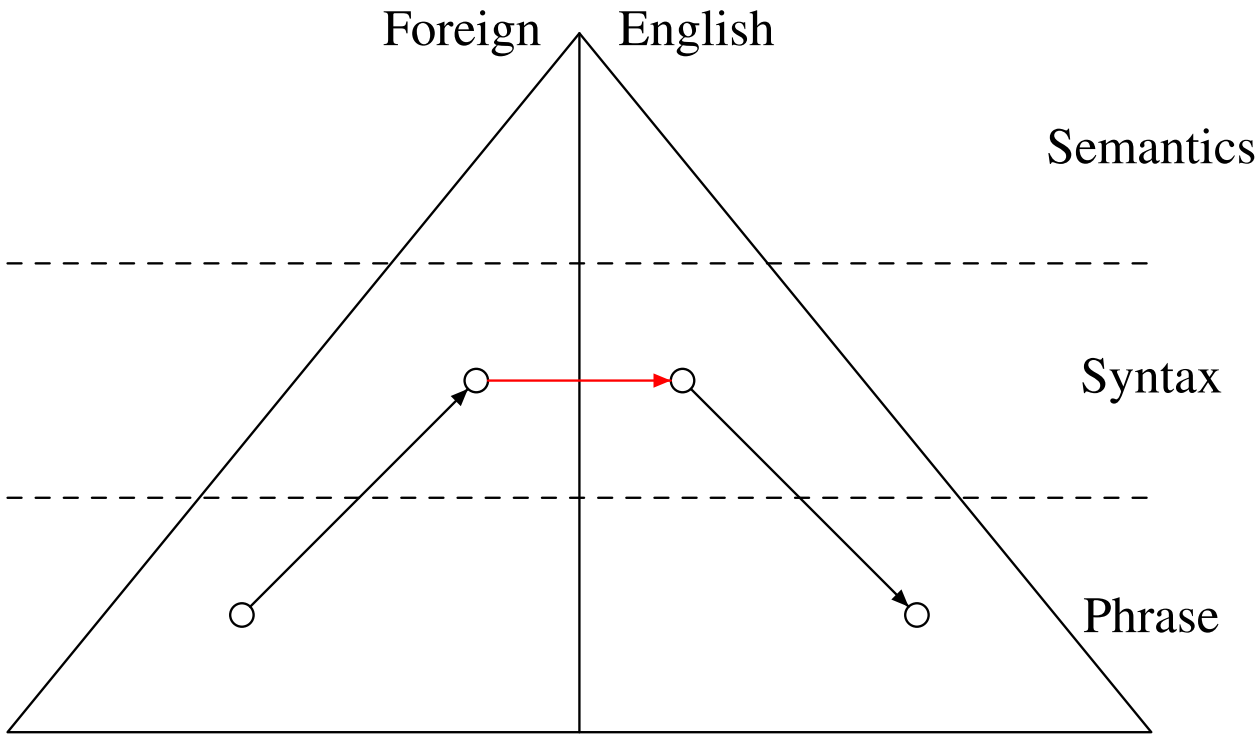
- that was before **statistics**
- small-scale, hand-crafted
- people said the same about syntax-based MT and look where it's now!

Different MT paradigms



phrase-based MT: n-grammatical

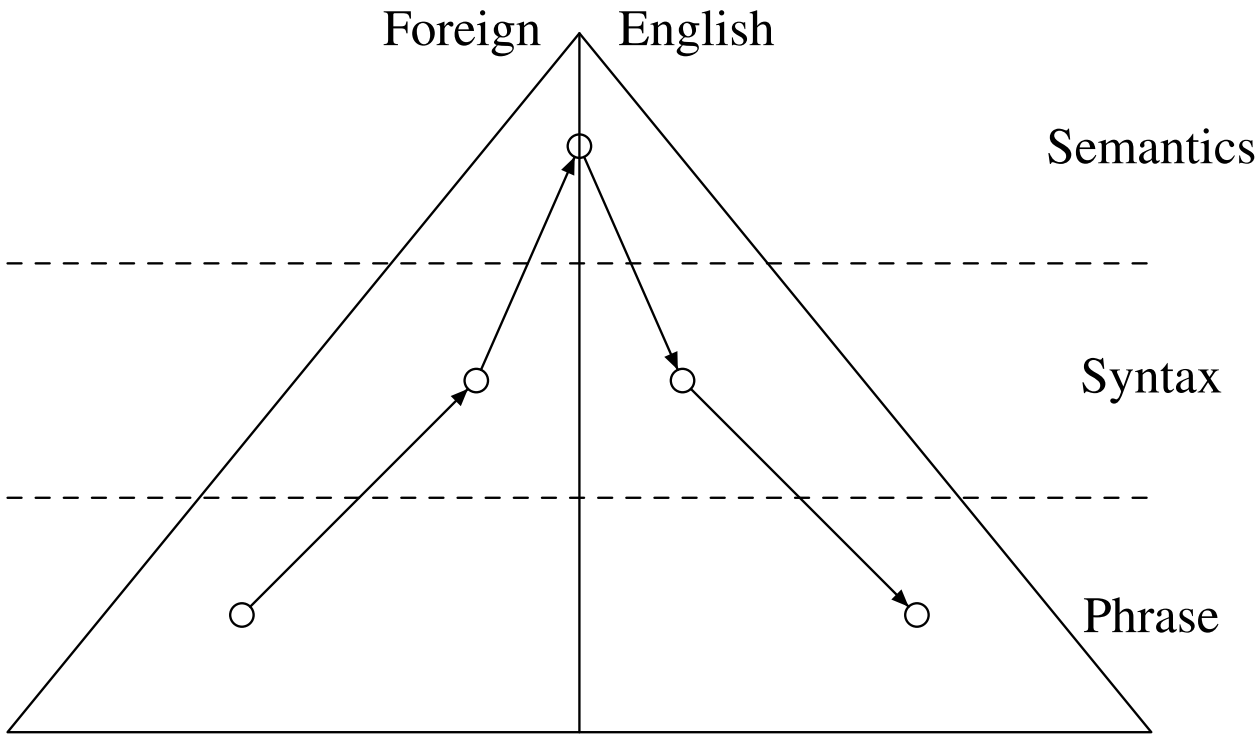
Different MT paradigms



phrase-based MT: n-grammatical

syntax-based MT: grammatical

Different MT paradigms



phrase-based MT: n-grammatical

syntax-based MT: grammatical

semantics-based MT: **sensible** and grammatical

Building an NLP system

With the theoretical background, it should be possible to carry out the same program that worked for syntax-based MT:

Building an NLP system

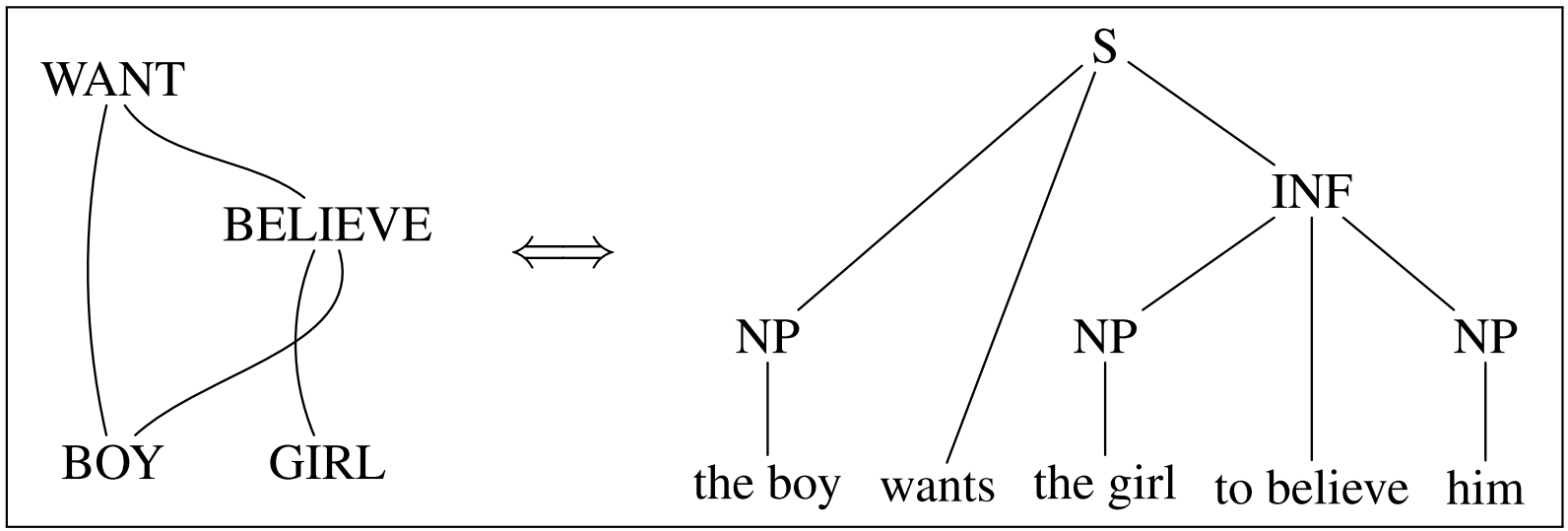
With the theoretical background, it should be possible to carry out the same program that worked for syntax-based MT:

- Collect lots of training data

Building an NLP system

With the theoretical background, it should be possible to carry out the same program that worked for syntax-based MT:

- Collect lots of training data

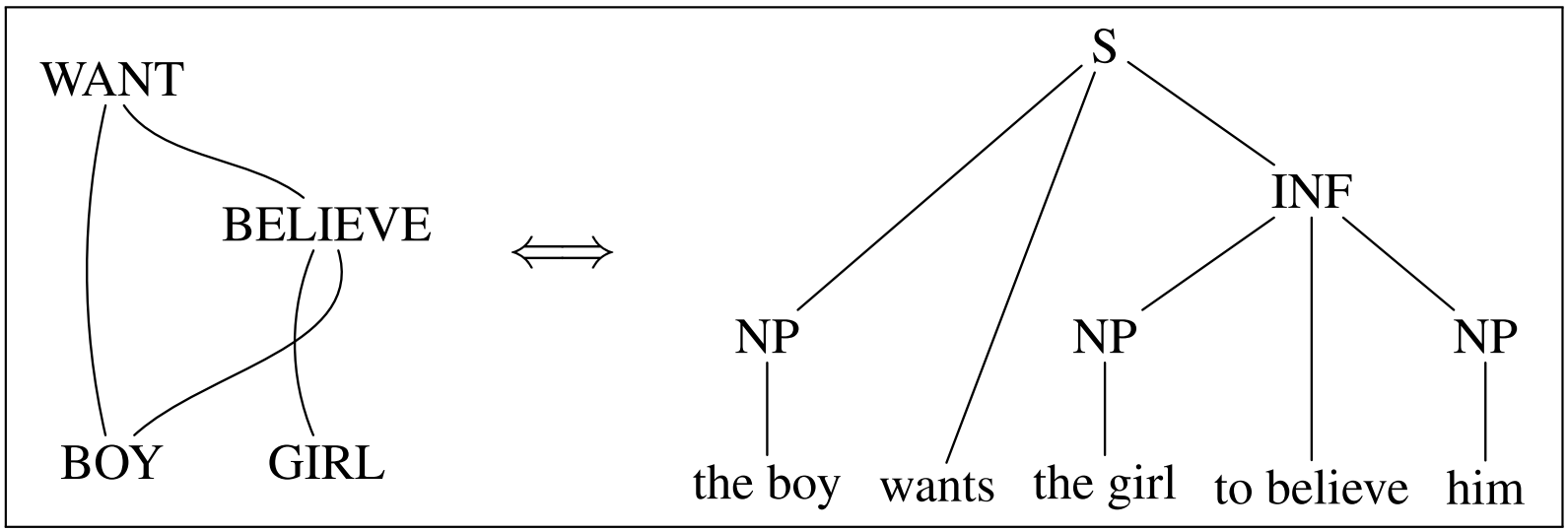


- Train models for parts of the translation pipeline

Building an NLP system

With the theoretical background, it should be possible to carry out the same program that worked for syntax-based MT:

- Collect lots of training data



- Train models for parts of the translation pipeline
- Use them in a bucket-brigade approach or in an integrated decoder

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

- packing the **set of derivations** into a dag acceptor
- packing the **set of output trees** into an RTG

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

- packing the **set of derivations** into a dag acceptor
- packing the **set of output trees** into an RTG

- unweighted and weighted ***n*-best generation**

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

- packing the **set of derivations** into a dag acceptor
- packing the **set of output trees** into an RTG

- unweighted and weighted ***n*-best generation**

- **backward application** (tree to dag)

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

- packing the **set of derivations** into a dag acceptor
- packing the **set of output trees** into an RTG

- unweighted and weighted ***n*-best generation**
- **backward application** (tree to dag)
- product construction: **intersection and union**

Toolkit (FSMNLP 2012)

We implemented (in Python):

- unweighted and weighted **membership checking**
- unweighted and weighted **dag-to-tree transductions**

- packing the **set of derivations** into a dag acceptor
- packing the **set of output trees** into an RTG

- unweighted and weighted ***n*-best generation**
- **backward application** (tree to dag)
- product construction: **intersection and union**
- **visualization** of trees and graphs using GraphViz

Future work

- Rule extraction
- Training

Future work

- Rule extraction
- Training
- Composition with tree transducers

Future work

- Rule extraction
- Training
- Composition with tree transducers
- Horizontal processing and sparse structures

Future work

- Rule extraction
- Training
- Composition with tree transducers
- Horizontal processing and sparse structures
- Annotated gold-standard data

Future work

- Rule extraction
- Training
- Composition with tree transducers
- Horizontal processing and sparse structures
- Annotated gold-standard data
- . . .

Future work

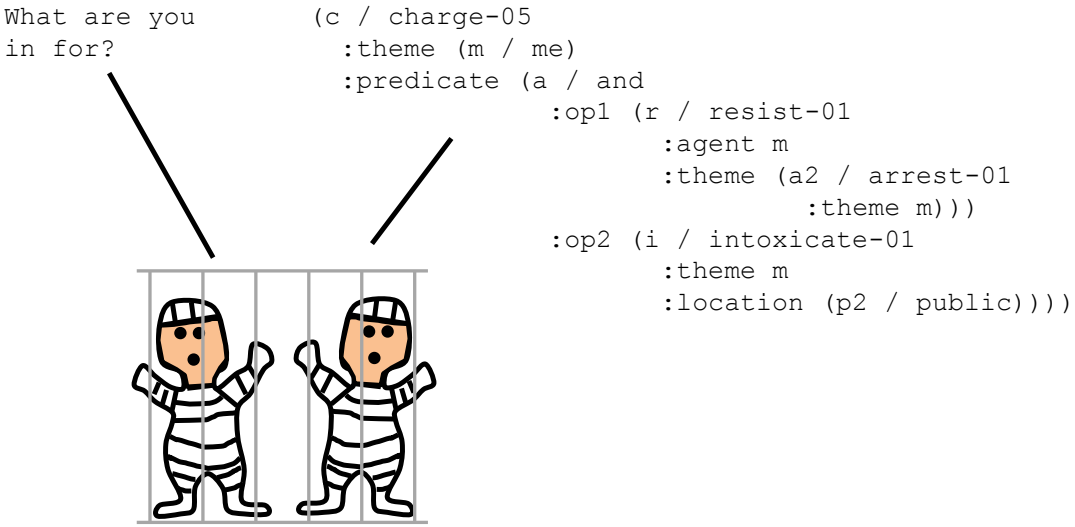
- Rule extraction
- Training
- Composition with tree transducers
- Horizontal processing and sparse structures
- Annotated gold-standard data
- ...

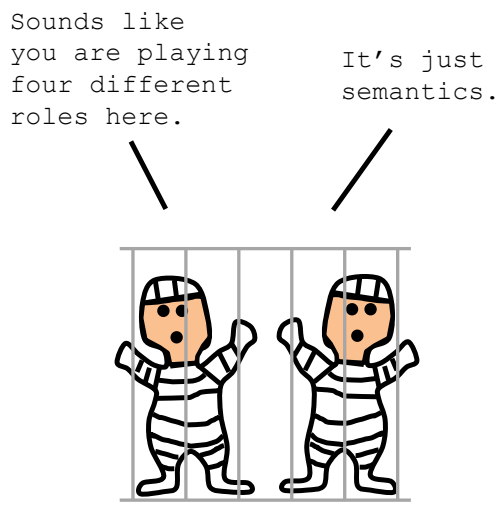
Download

`http://www.ims.uni-stuttgart.de/~daniel/dagger/`

The ~~end~~ beginning

Thank you for your attention! – Questions?





References

Tsutomu Kamimura and Giora Slutzki.
1981.
Parallel and two-way automata on directed ordered acyclic graphs.
Inf. Control, 49(1):10–51.

Tsutomu Kamimura and Giora Slutzki.
1982.
Transductions of dags and trees.
Math. Syst. Theory, 15(3):225–249.