Overview

Vilem Mathesius Lecture Series 19
March, 2004

Stuart M. Shieber
Division of Engineering and Applied Sciences
Harvard University

Computational Linguistics

The study of human language
using tools and techniques of computer science,
with possible application to its automated processing

Division of Engineering and Applied Sciences
Harvard University

Four Strands of Computational Linguistics

1. Science: Applying computational insights to the study of language
   - formalism
   - rigor
   - process

Four Strands of Computational Linguistics

1. Science: Applying computational insights to the study of language
2. Engineering: Construction or improvement of computational artifacts to solve user problems involving language
   - speech recognition
   - machine translation
   - natural language interfaces to software systems
   - information retrieval or extraction

Four Strands of Computational Linguistics

1. Science: Applying computational insights to the study of language
2. Engineering: Construction or improvement of computational artifacts to solve user problems involving language
3. Tool-building: Construction or improvement of tools to enable (1) or (2)
   - weighted finite-state transducers
   - synchronous tree-adjoining grammars

Four Strands of Computational Linguistics

1. Science: Applying computational insights to the study of language
2. Engineering: Construction or improvement of computational artifacts to solve user problems involving language
3. Tool-building: Construction or improvement of tools to enable (1) or (2)
4. *Philosophy: Applying computational insights to philosophical questions about language*

Lecture 1:

Resurrecting the Turing Test

Division of Engineering and Applied Sciences
Harvard University

The Goal

Alan Turing, Computing Machinery and Intelligence, 1950:

- Can machines think?
- How could we tell?

Flashback to the 17th century:

- Do animals have souls?
- How could we tell?

Descartes on the Singularity of Verbal Behavior

"In fact, none of our external actions can show anyone who examines them that our body is not just a self-moving machine but contains a soul with thoughts, *with the exception of spoken words*, or other signs that have relevance to particular topics without expressing any passion."

— René Descartes
Letter to the Marquess of Newcastle, 1646

Division of Engineering and Applied Sciences
Harvard University

Cordemoy on Indistinguishability

"That if the Bodies, which are like mine, had nothing but the facilness of pronouncing Words, I should not therefore believe that they had the advantage of being united to Souls: But then, if I *finde by all the Experiments, I am capable to make, that they use speech as I do*, I shall think, I have infallible reason to believe that they have a soul as I."

— Gèraud de Cordemoy
A philosophicall discourse concerning speech, 1668

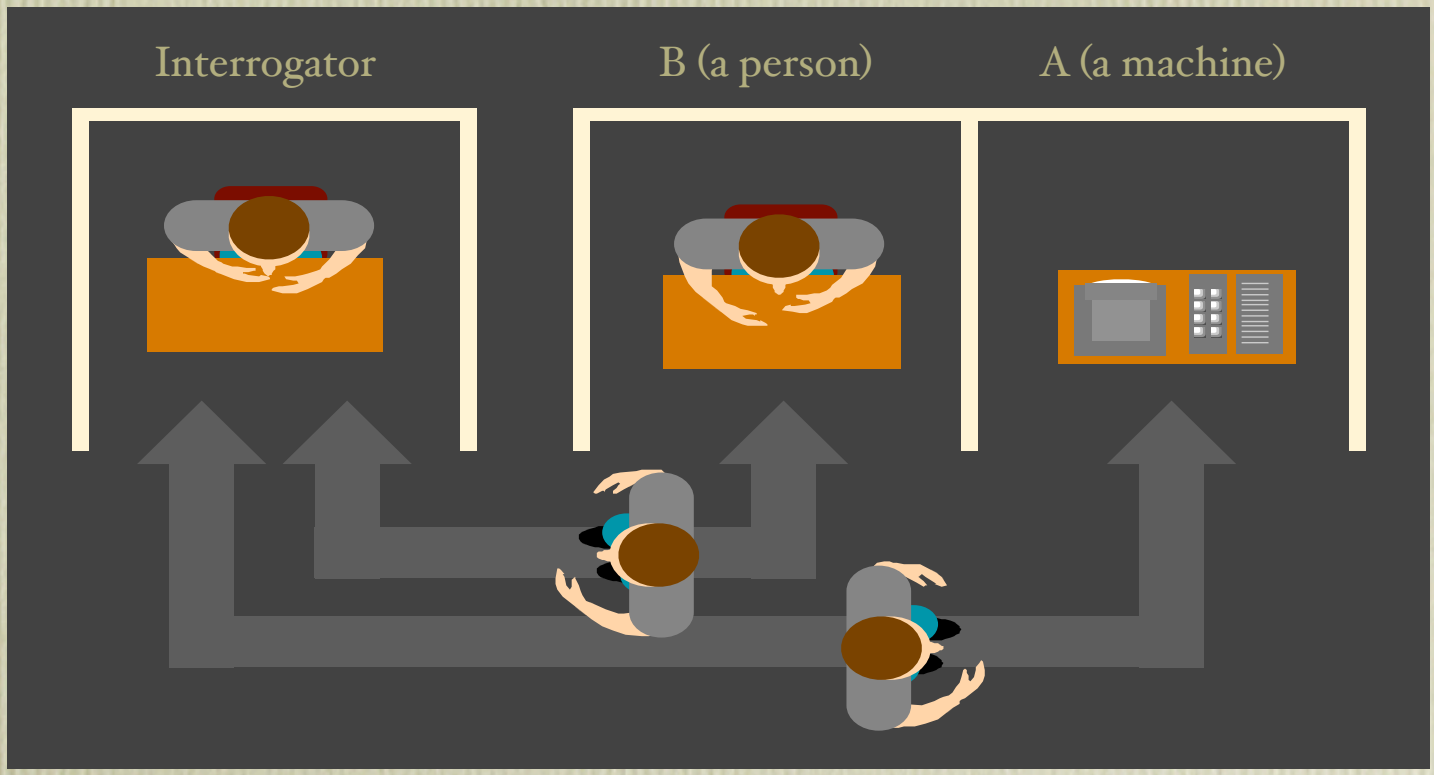
Division of Engineering and Applied Sciences
Harvard University

Turing's Agenda

"*I propose to consider the question, 'Can machines think?'* This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. *Instead of attempting such a definition I shall replace the question by another*, which is closely related to it and is expressed in relatively unambiguous words."

— Alan Turing
Computing machinery and intelligence, 1950

Turing's "Imitation Game"



Necessary versus Sufficient Conditions

Is ability to pass a Turing Test a necessary condition of intelligence?

- "May not machines carry out something which ought to be described as thinking but which is very different from what a man does? This objection is a very strong one, but at least we can say that if, nevertheless, a machine can be constructed to play the imitation game satisfactorily, we need not be troubled by this objection." — Turing, 1950

Is ability to pass a Turing Test a sufficient condition of intelligence?

Turing:
Replace the Question

"Instead of attempting such a definition I shall replace the question by another. ... *The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion.*"

— Turing, 1950

"If Turing intends that the question of the success of the machine at the imitation game replace the question about machines thinking, then *it is difficult to understand how we are to judge the propriety and adequacy of the replacement if the question being replaced is too meaningless to deserve discussion*. Our potential interest in the imitation game is aroused not by the fact that a computer might learn to play yet another game, but that in some way this test reveals a connection between possible computer activities and our ordinary concept of human thinking."

— Moor, 1976

Current Views

The Turing Test is not a sufficient condition for intelligence.

- Gunderson — flexibility of behavior
- Davidson — semantics
- Searle — intentionality
- Block — richness of information processing
- ...

The Turing Test is a sufficient condition for intelligence.

- Dennett

Dennett:
The Turing Test is Sufficient

"The Turing test in unadulterated, unrestricted form, as Turing presented it, is plenty strong if well used. I am confident that no computer in the next twenty years is going to pass the unrestricted Turing test. They may well win the World Chess Championship or even a Nobel Prize in physics, but they won't pass the unrestricted Turing test. Nevertheless, it is not, I think, impossible in principle for a computer to pass the test, fair and square. I'm not running one of those a priori 'computers can't think' arguments. *I stand unabashedly ready, moreover, to declare that any computer that actually passes the unrestricted Turing test will be, in every theoretically interesting sense, a thinking thing.*"

— Dennett, 1985

Block: The Turing Test is Insufficient

I conclude that the capacity to emit sensible responses is not sufficient for intelligence, and so the *neo-Turing Test conception of intelligence is refuted* (along with the older and cruder Turing Test conceptions). I also conclude that whether behavior is intelligent behavior is in part a matter of how it is produced. Even if a system has the actual and potential behavior characteristic of an intelligent being, if its internal processes are like those of the machine described, it is not intelligent.

— Block, 1981

The Turing Syllogism

If an agent passes a Turing Test, then it produces a sensible sequence of verbal responses to a sequence of verbal stimuli.

The Turing Syllogism

If an agent passes a Turing Test, then it produces a sensible sequence of verbal responses to a sequence of verbal stimuli.

If an agent produces a sensible sequence of verbal responses to a sequence of verbal stimuli, then it is intelligent.

The Turing Syllogism

If an agent passes a Turing Test, then it produces a sensible sequence of verbal responses to a sequence of verbal stimuli.

If an agent produces a sensible sequence of verbal responses to a sequence of verbal stimuli, then it is intelligent.

Therefore, if an agent passes a Turing Test, then it is intelligent.

The Turing Syllogism

If an agent passes a Turing Test, then it produces a sensible sequence of verbal responses to a sequence of verbal stimuli.

The Turing Test conception of intelligence:
If an agent produces a sensible sequence of verbal responses to a sequence of verbal stimuli, then it is intelligent.

Therefore, if an agent passes a Turing Test, then it is intelligent.

The Occasional Conception

If an agent produces a sensible sequence of verbal responses to a sequence of verbal stimuli, then it is intelligent.

The Capacity Conception

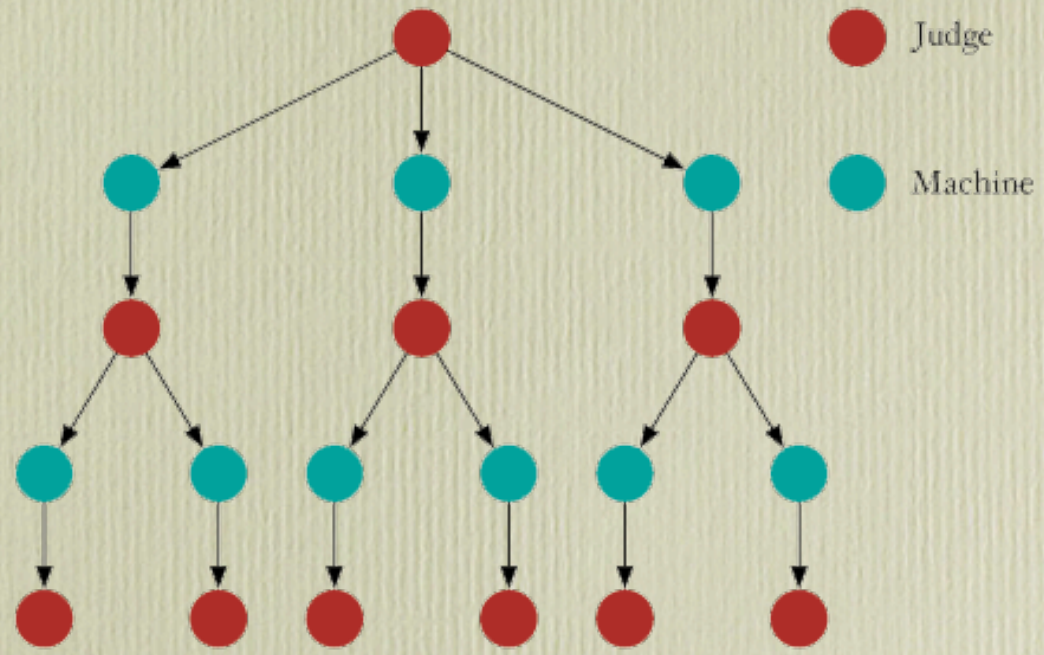
If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli, *whatever they may be*, then it is intelligent.

Promissory Note

The Turing Test needs to demonstrate

- *Capacity* a general capacity for sensible response to verbal stimuli

The Aunt Bertha Machine



Division of Engineering and Applied Sciences
Harvard University

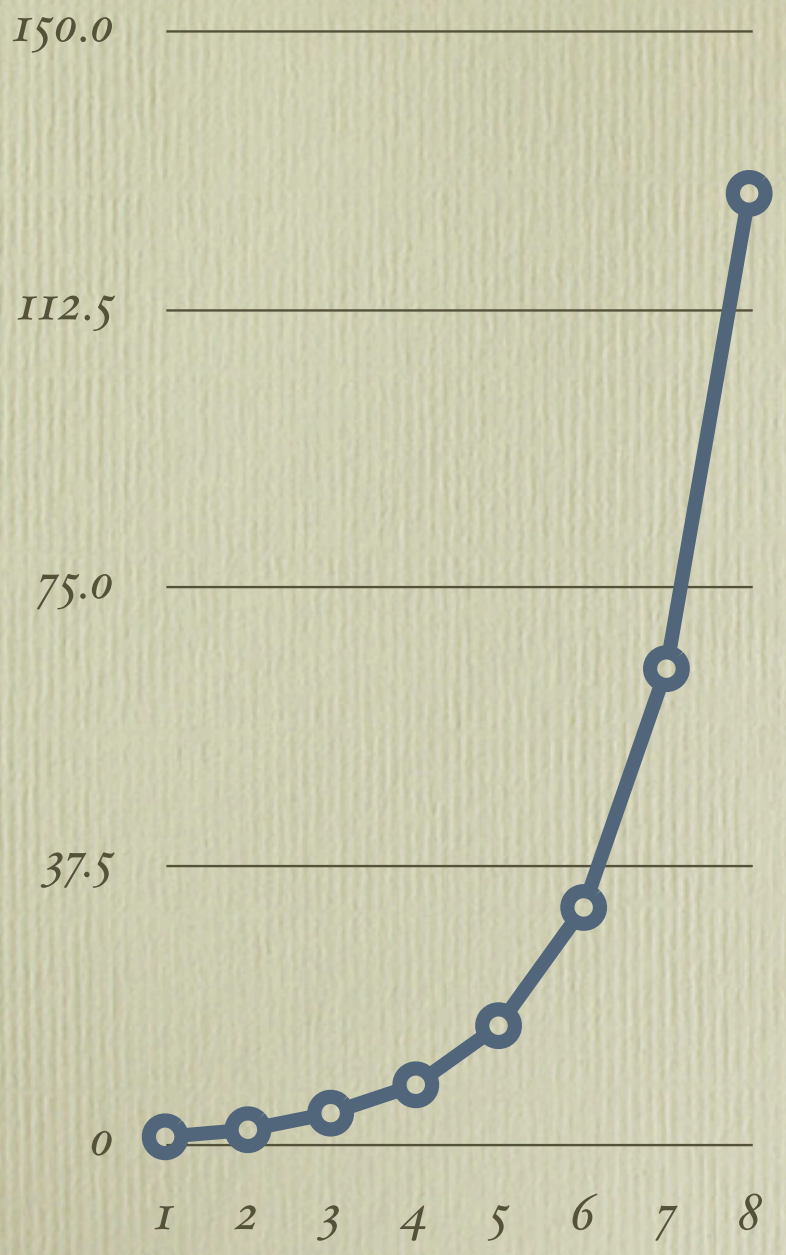
Exponential Explosion

[Intelligence is the capacity to emit sensible sequences of responses to stimuli, so long as this is accomplished in a way that averts exponential explosion of search. ... Dennett tells me that he advocates [this conception]. (Block, 1981)]

The Compact Conception

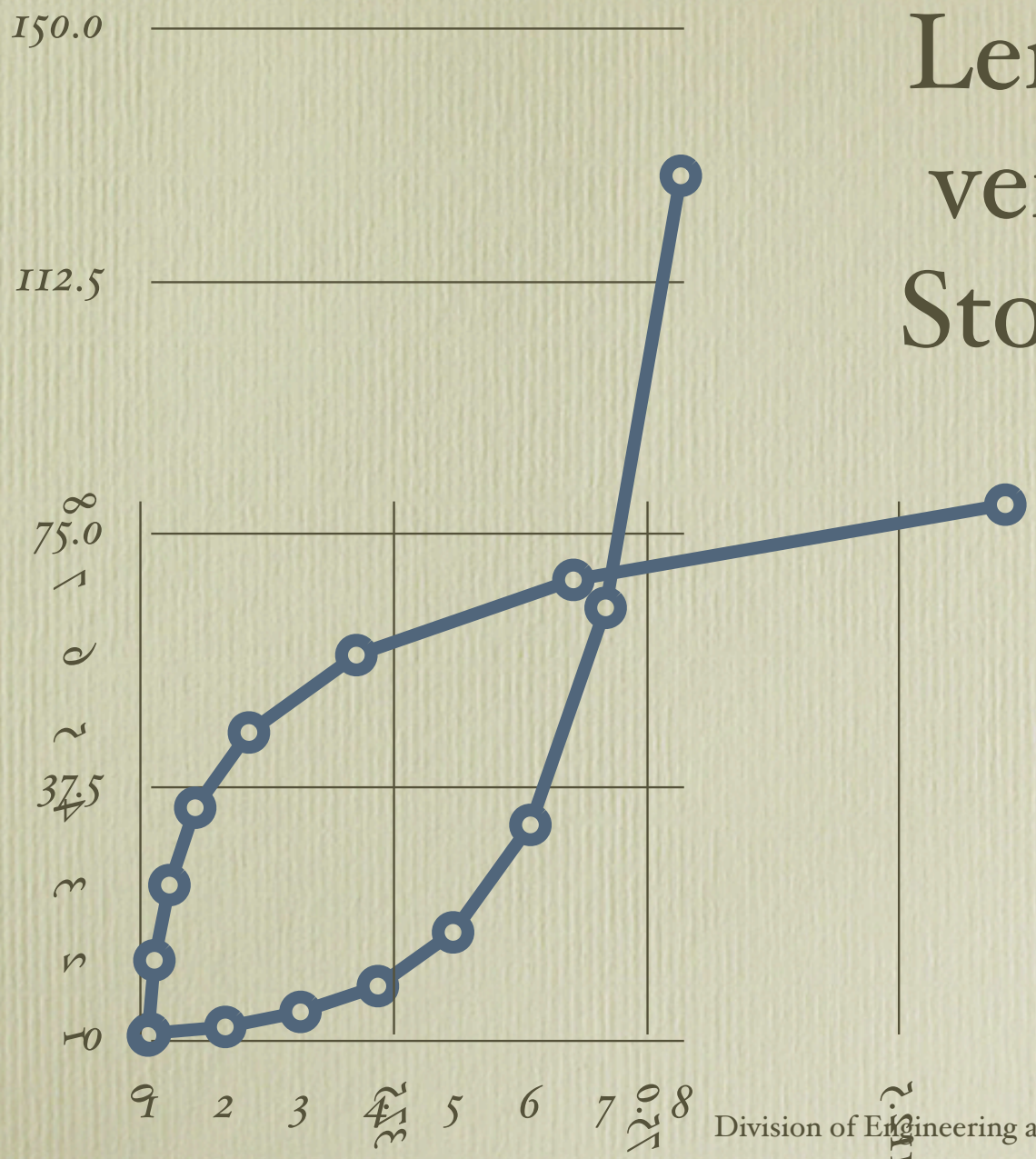
> [Intelligence is the capacity to emit sensible sequences of responses to stimuli, so long as this is accomplished in a way that averts exponential explosion of search. ... Dennett tells me that he advocates [this conception]. (Block, 1981)]

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli, whatever they may be, and *without requiring exponential storage*, then it is intelligent.



Storage versus Length

Length versus Storage



The Compact Conception

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli, whatever they may be, and *without requiring exponential storage*, then it is intelligent.

Division of Engineering and Applied Sciences
Harvard University

Modified The Compact Conception

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli, whatever they may be, and *without requiring exponential storage*, then it is intelligent.

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli *that is of length at least logarithmic in the storage capacity of the agent*, whatever they may be, then it is intelligent.

Division of Engineering and Applied Sciences
Harvard University

Promissory Notes

The Turing Test needs to demonstrate

- *Capacity* a general capacity for sensible response to verbal stimuli
- *Complexity* of length at least logarithmic in the storage capacity of the subject under test.

Interactive Proofs

Proofs based on interaction and randomization.

Division of Engineering and Applied Sciences
Harvard University

Jelly Bean Counting

1. *P*[rover] examines the jar to obtain bean count b_0
2. *V*[erifier] privately removes *d* jelly beans (0 or 1, chosen randomly) and shakes the jar
3. *P* examines the jar to obtain bean count b_1
4. *P* reports $b_1 - b_0$
5. If $d = b_1 - b_0$, *V* accepts; otherwise *V* rejects

Properties of the Interactive Proof Protocol

Probabilistic proof condition

- No false rejections
- False acceptance 1/2 the time

Exponential confidence

- For k rounds, false acceptance $1/2^k$

Only verifier receives proof

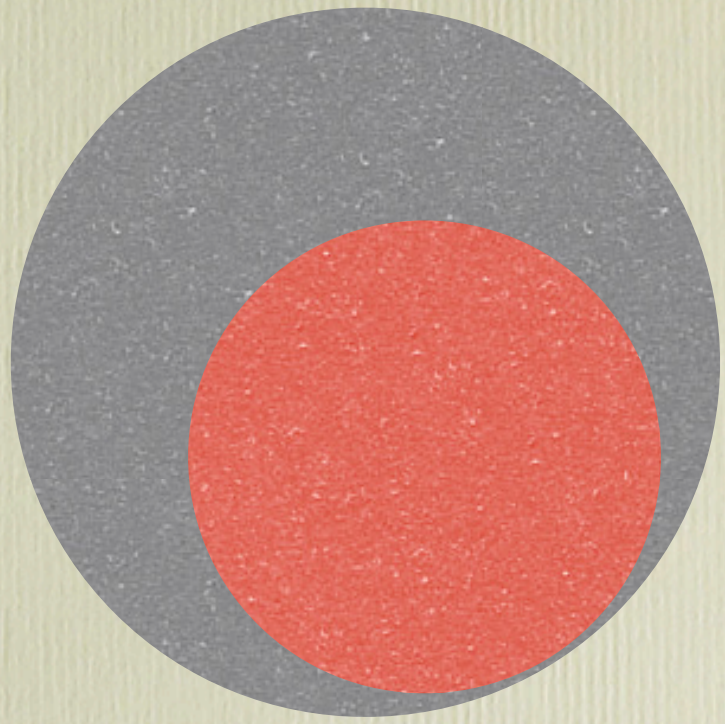
- Verifier can generate accepting protocols unilaterally

Interactive Proofs of a Capacity

- *P* has a capacity to compute *f* if *P* computes *f* correctly on a substantial fraction (t_l) of inputs

- Assume *P* computes *f* correctly on t_p of the inputs.

$t_p > t_l$?



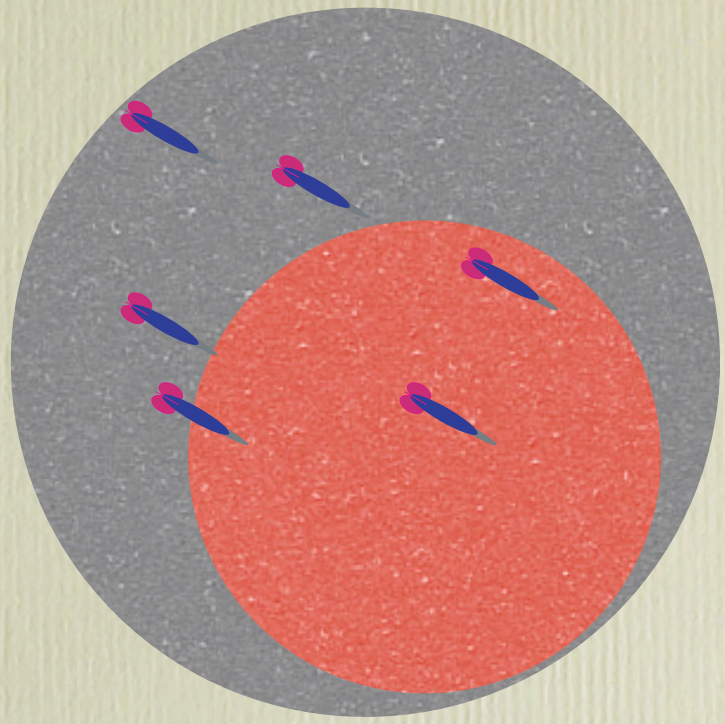
Division of Engineering and Applied Sciences
Harvard University

Interactive Proofs of a Capacity

- P has a capacity to compute f if P computes f correctly on a substantial fraction (t_l) of inputs

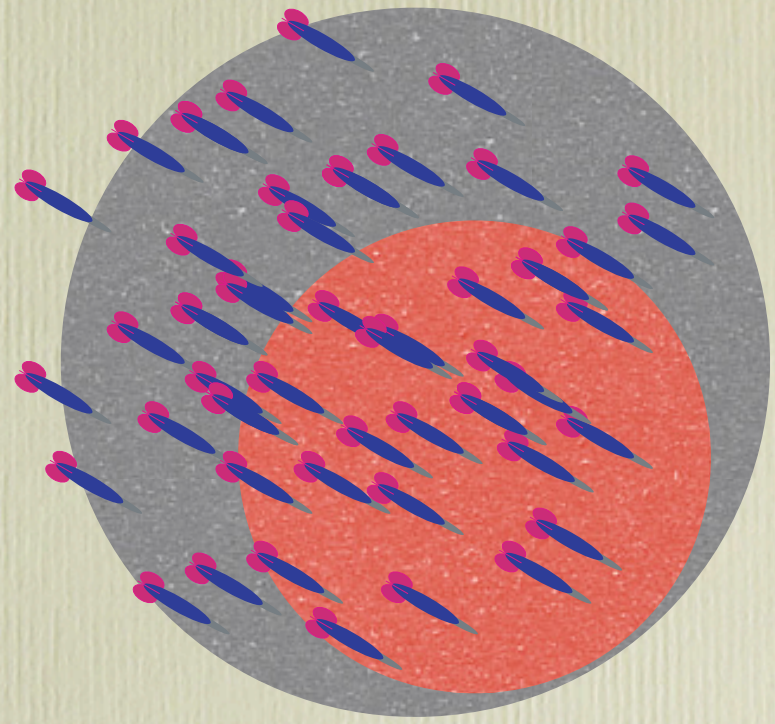
- Assume P computes f correctly on t_p of the inputs.

$t_p > t_l$?



Interactive Proofs of a Capacity

- P has a capacity to compute f if P computes f correctly on a substantial fraction (t_l) of inputs
- Assume P computes f correctly on t_p of the inputs.
- Assume P computes f correctly on t of the samples.



Interactive Proofs of a Capacity

P has a capacity to compute f if P computes f correctly on a substantial fraction (t_l) of inputs

Assume P computes f correctly on t_p of the inputs.

Assume P computes f correctly on t of the samples.

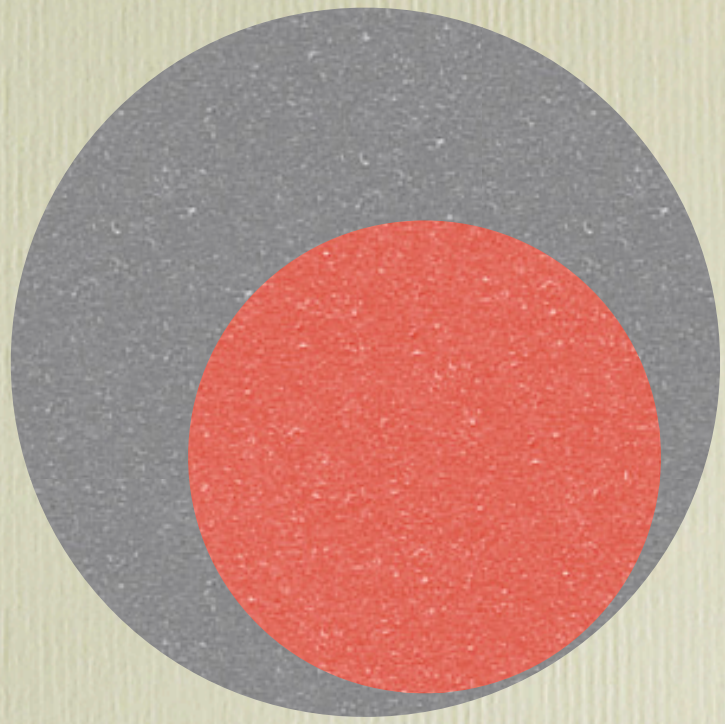
Let $t_s > t_l$ be a sample threshold.

Then,

$$\text{if } t_p < t_l, \Pr(t > t_s) < 1/c^k$$

Interactive Proofs of a Capacity

- $t_l = 1/2$
- $t_s = 3/4$
- $k = 300$ rounds
- If $t_p < 1/2$,
  $\Pr(t > 3/4) < 1/10^{10}$

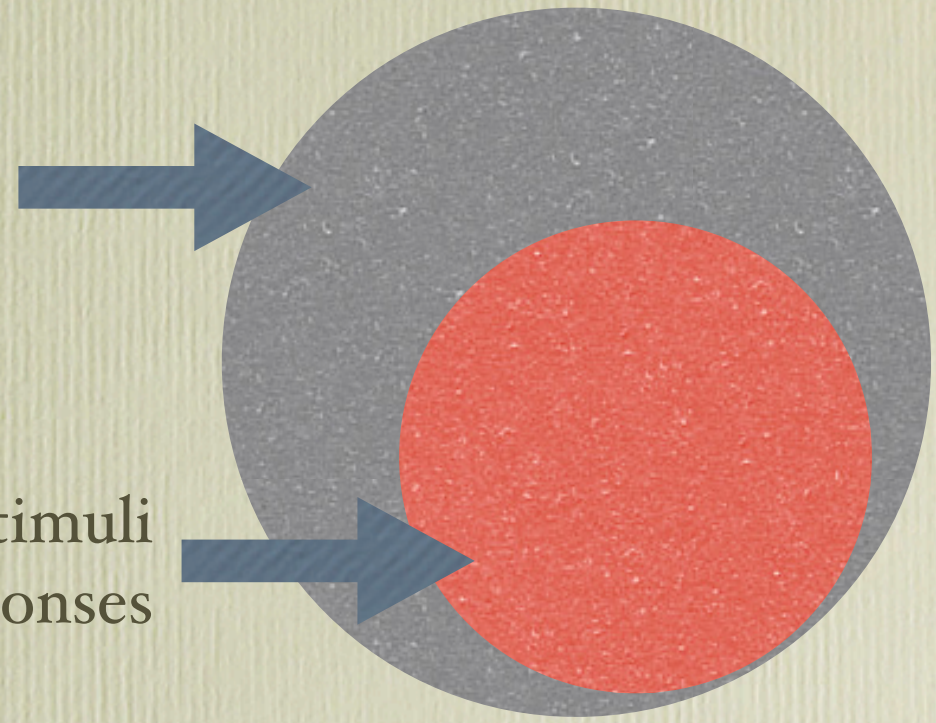


Division of Engineering and Applied Sciences
Harvard University

The Turing Test as an Interactive Proof

Space of all possible verbal stimuli

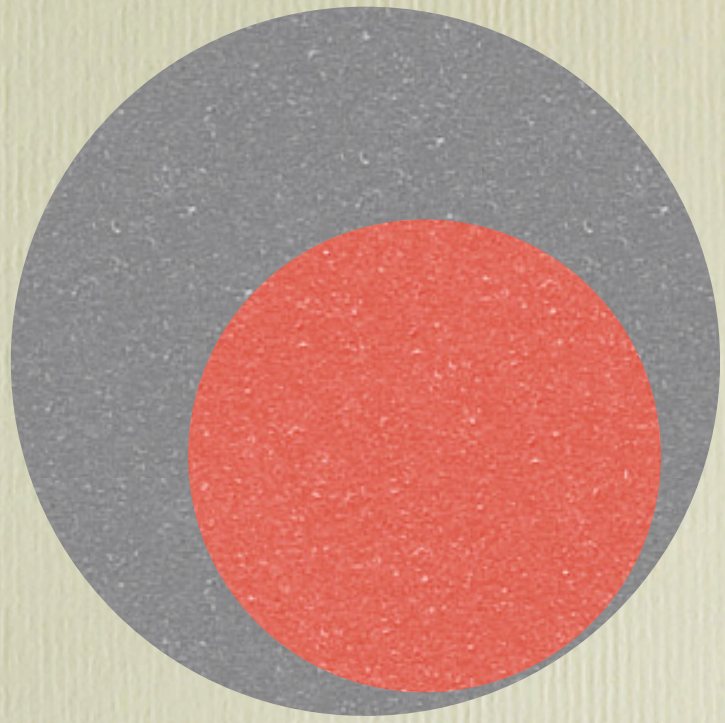
Subspace of verbal stimuli with sensible responses



Division of Engineering and Applied Sciences
Harvard University

The Turing Test as an Interactive Proof

Run k rounds of Turing Tests. Observe that sensible responses are generated on a fraction of inputs greater than the threshold t_s. Then, with probability of error exponentially small in k, sensible responses are generated on at least a fraction t_l of all inputs.



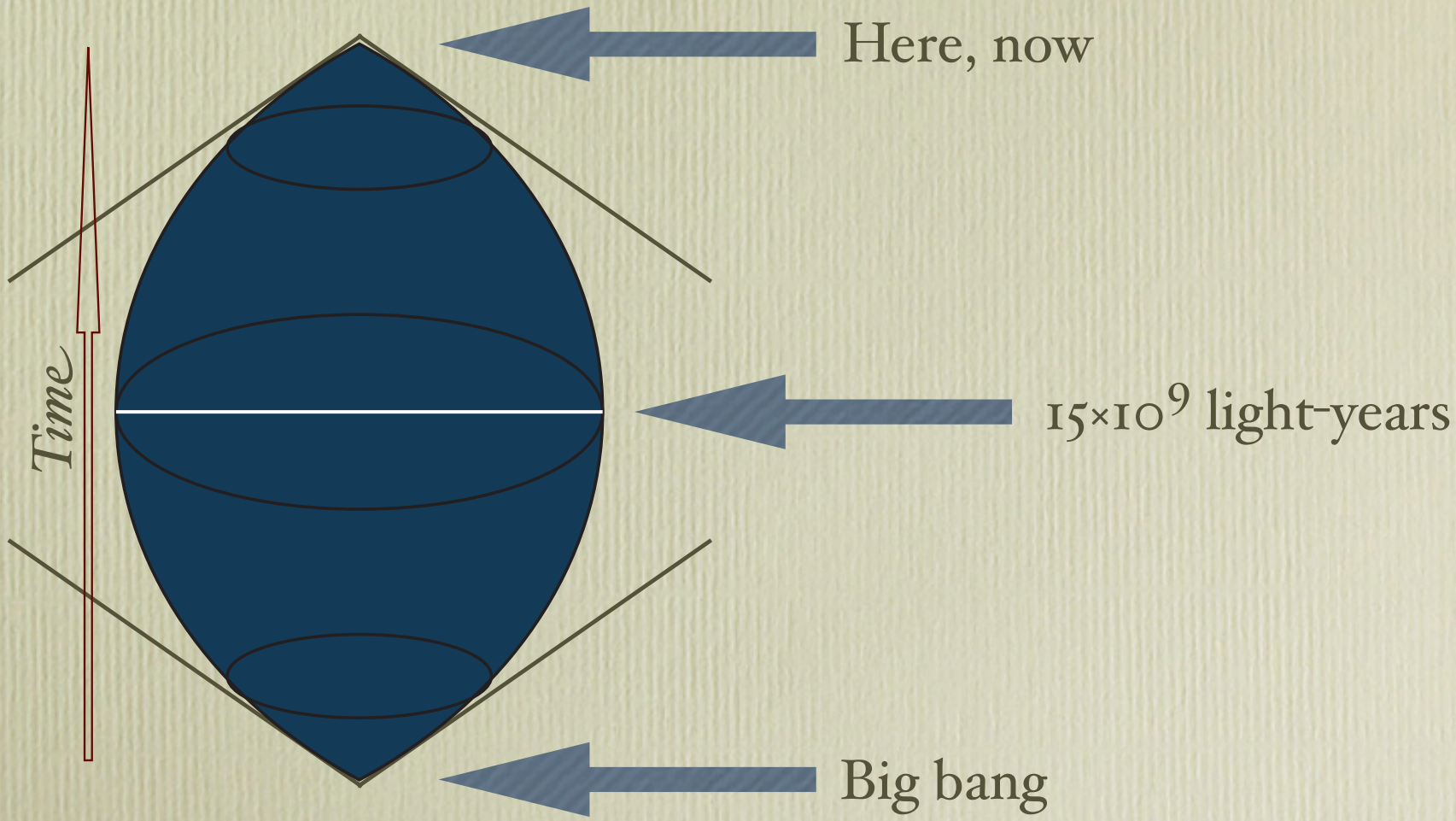
Division of Engineering and Applied Sciences
Harvard University

Promissory Notes

The Turing Test needs to demonstrate

- *Capacity* a general capacity for sensible response to verbal stimuli
- *Complexity* of length at least logarithmic in the storage capacity of the subject under test.

Size of the Universe



Division of Engineering and Applied Sciences
Harvard University

Storage Capacity of the Universe

Volume: $(15\times10^{9}$ light-years$)^3 = (15\times10^{9}\times10^{16}$ meters$)^3$

Density: 1 bit per $(10^{-35}$ meters$)^3$

Total storage capacity: 10^{184} bits $< 10^{200}$ bits $\cong$ 2^{670} bits

Critical Turing Test length: 670 bits $\cong$ 670 characters $\cong$ 140 words $<$ **1 minute**

Division of Engineering and Applied Sciences
Harvard University

The Turing Syllogism

If an agent passes a Turing Test, then it produces a sensible sequence of verbal responses to a sequence of verbal stimuli.

If an agent produces a sensible sequence of verbal responses to a sequence of verbal stimuli, then it is intelligent.

Therefore, if an agent passes a Turing Test, then it is intelligent.

Modified The Compact Conception

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli, whatever they may be, and *without requiring exponential storage*, then it is intelligent.

If an agent has *the capacity* to produce a sensible sequence of verbal responses to a sequence of verbal stimuli *that is logarithmic in the storage capacity of the agent*, whatever they may be, then it is intelligent.

Division of Engineering and Applied Sciences
Harvard University

The Turing Syllogism

If an agent *passes k rounds of a Turing Test of at least one minute in length*, then *with probability of error exponentially small in k*, it has the capacity to produce a sensible sequence of verbal responses to a sequence of verbal stimuli that is logarithmic in the storage capacity of the agent, whatever they may be.

The modified compact conception of intelligence:

If an agent has the capacity to produce a sensible sequence of verbal responses to a sequence of verbal stimuli that is logarithmic in the storage capacity of the agent, whatever they may be, then it is intelligent.

Therefore, if an agent passes a Turing Test, then it is intelligent.

Division of Engineering and Applied Sciences
Harvard University

Summary

The Turing Test can be most profitably viewed as an *interactive proof of intelligence*:

- In form
- In performance

Who is right?

- Classical notion of proof: Block is right
- Interactive notion of proof: Dennett is right

Four Strands of Computational Linguistics

1. Science: Applying computational insights to the study of language
2. Engineering: Construction or improvement of computational artifacts to solve user problems involving language
3. *Tool-building: Construction or improvement of tools to enable (1) or (2)*
4. Philosophy: Applying computational insights to philosophical questions about language

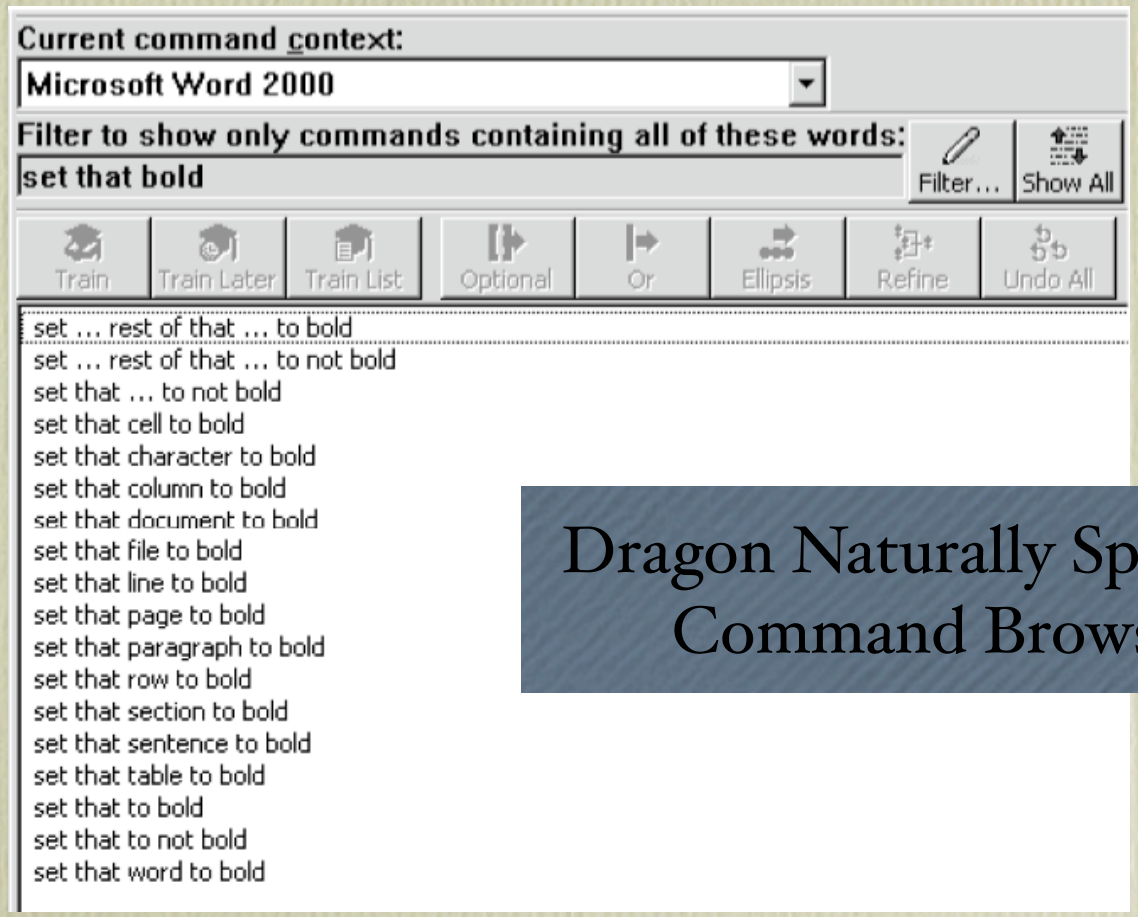
Lecture 2:

Towards a Universal Natural-Language Pipeline

Stuart M. Shieber

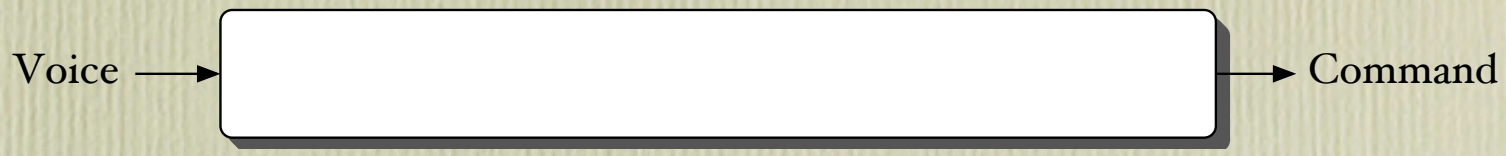
Division of Engineering and Applied Sciences
Harvard University

Motivation

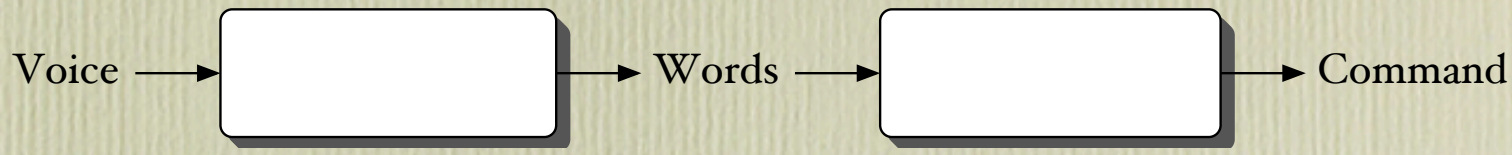


Dragon Naturally Speaking Command Browser

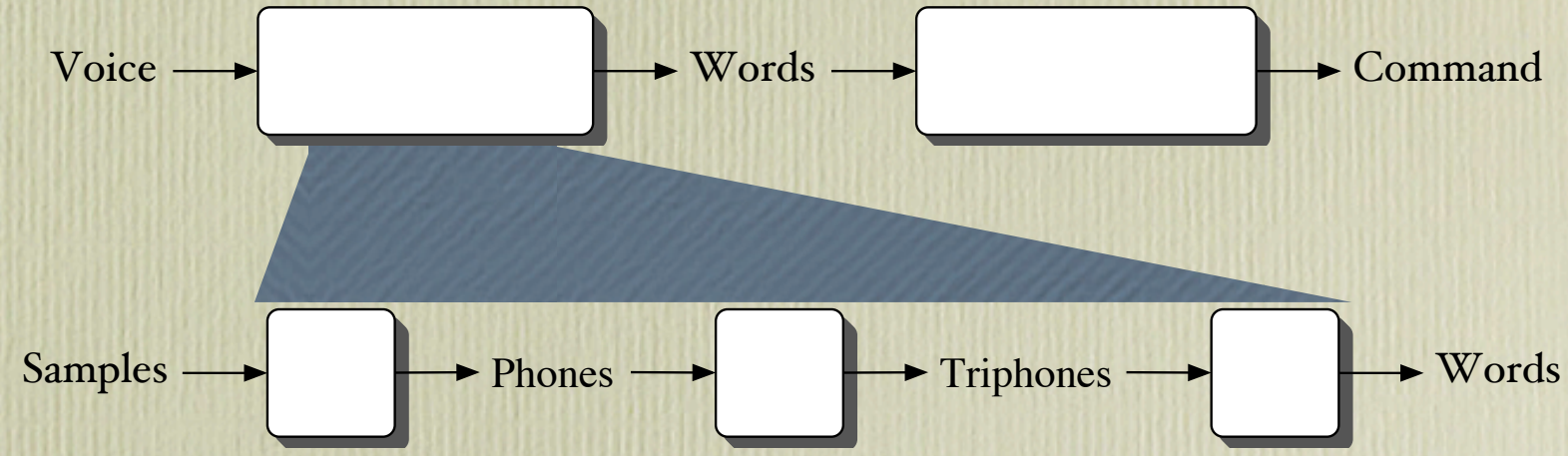
Voice to Commands



Voice to Commands



Voice to Commands



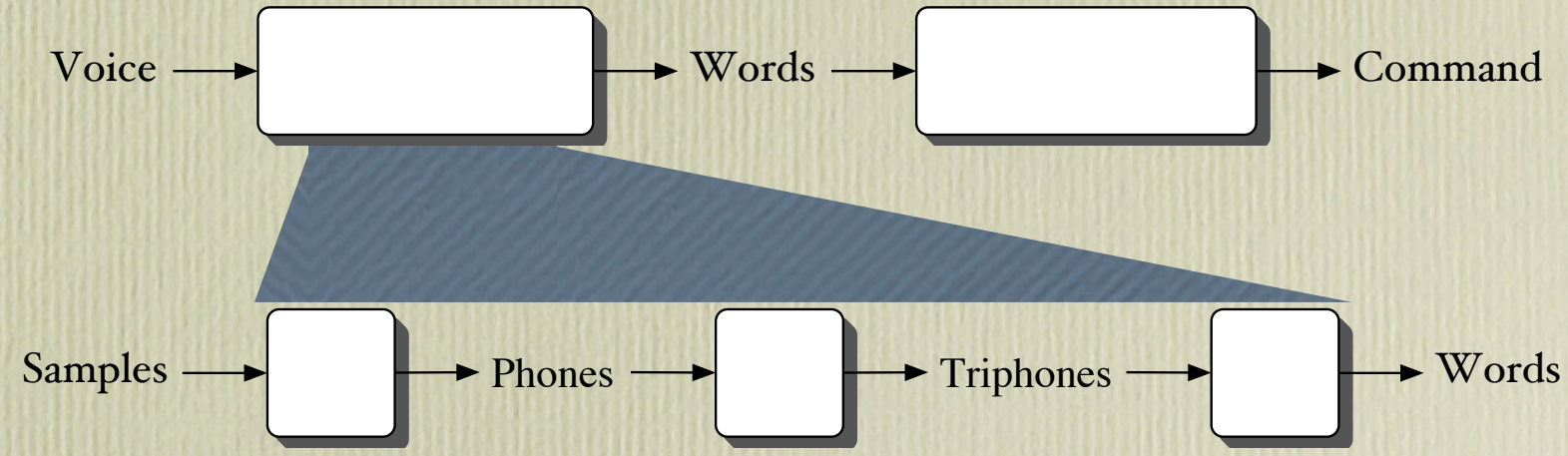
Hidden Markov Models

phonotactic models

dictionaries

language models

Voice to Commands



Hidden Markov Models

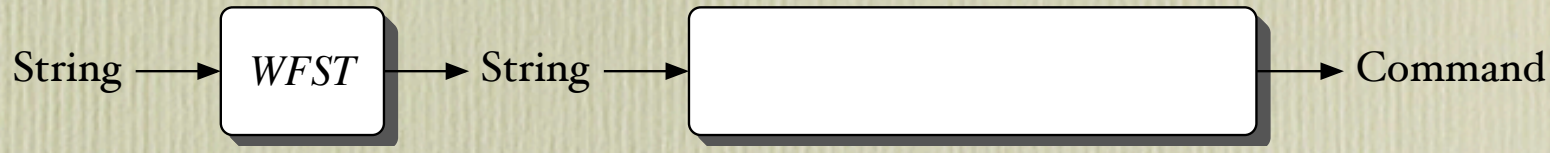
phonotactic models

Weighted Finite State Transducers

dictionaries

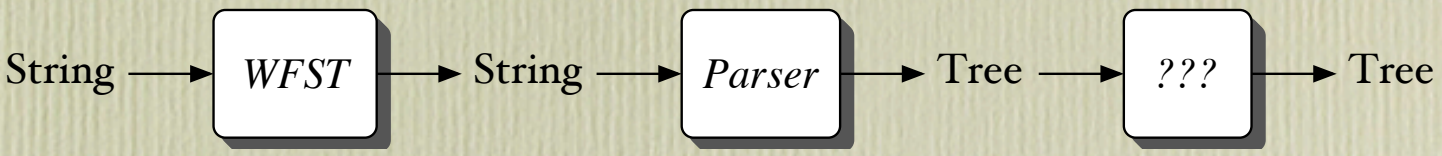
language models

Voice to Commands



Voice to Commands

The Universal NL Pipeline



Strings is to *WFST* as *trees* is to ***???***

Ubiquity of Tree Transformation

Command interpretation

- Database query construction

Semantic interpretation

- Semantic disambiguation

Machine translation transfer

Parsed corpus manipulation and normalization

Natural language generation

- Logical form canonicalization

Overview

Finite-state automata and transducers

Weighted automata and transducers

Tree transducers (and their insufficiency)

Bimorphisms and synchronous grammars

Regular Languages and Finite-State Automata

Division of Engineering and Applied Sciences
Harvard University

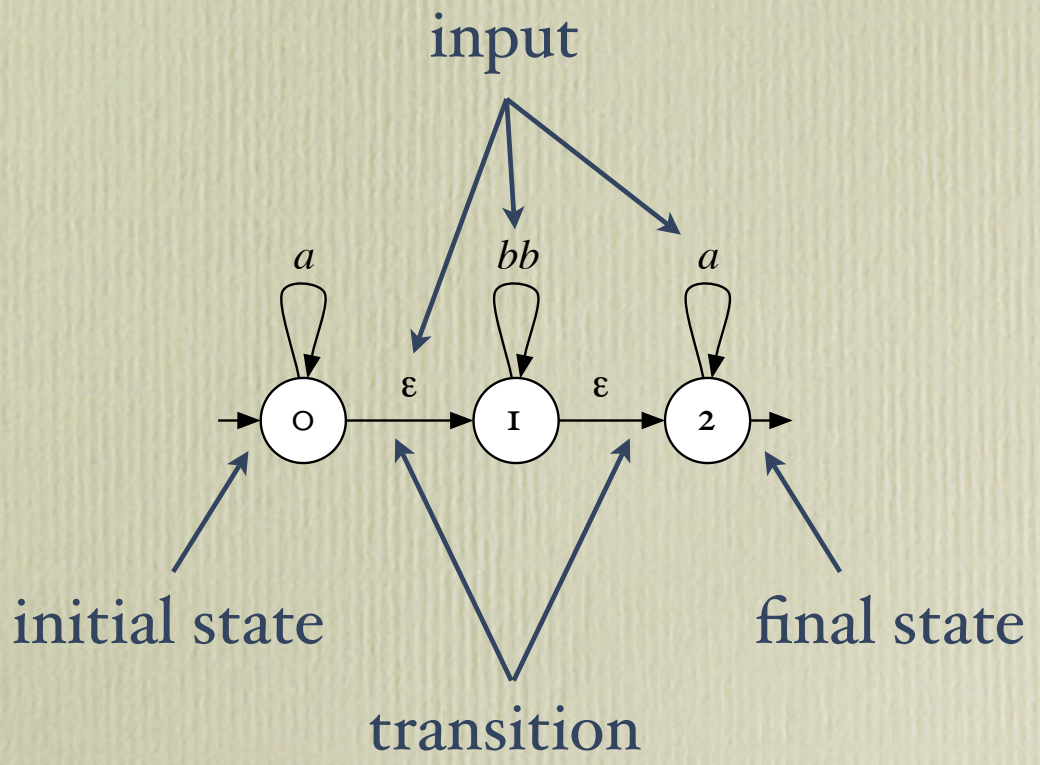
Regular Languages

A language is a set of strings.

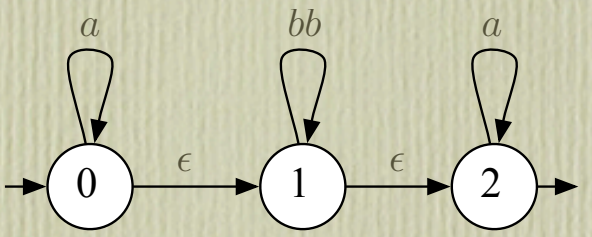
Regular languages is the smallest class of languages including

- the empty language
- singleton languages
- closure under
  - union $(L_1 \cup L_2)$
  - concatenation $(L_1 \cdot L_2)$
  - iteration closure (L^*)

Finite-State Automata



Finite-State Automata



A derivation:

o *aaabba#* → o *aabba#* → o *abba#* → o *bba#*

→ 1 *bba#* → 1 *a#* → 2 *a#* → 2 *#* → *#*

Definition of recognition:

- accept *w* if and only if o *w#* →* *#*

Degrees of Freedom

Lots of things make no difference:

- Granularity
- Epsilon removal
- Left-right reversal
- Determinization
- Minimization

Finite-State Transducers

Division of Engineering and Applied Sciences
Harvard University

Regular Relations

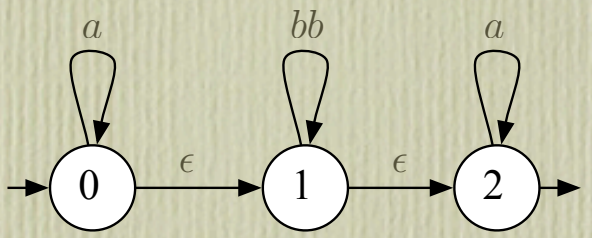
A string relation is a set of *pairs of* strings.

$$\textit{input} : \textit{output}$$

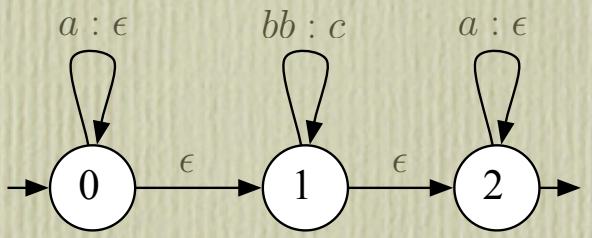
Regular relations is the smallest class of relations including:

- the empty language
- singleton languages, e.g., $\{a : \epsilon\}, \{\epsilon : a\}, \ldots$
- closure under
  - union
  - concatenation
  - iteration closure

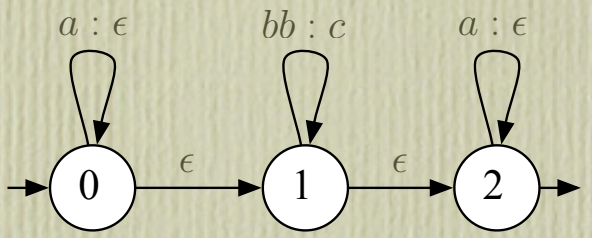
Sample FST



Sample FST



Sample FST



A derivation:

○ *aaabba#* → ○ *aabba#* → ○ *abba#* → ○ *bba#*

→ 1 *bba#* → *c* 1 *a#* → *c* 2 *a#* → *c* 2 *#* → *c #*

Definition of recognition:

- accept *s*:*t* if and only if ○ *s #* →* *t #*

Degrees of Freedom

granularity of input and output

left-right reversal

epsilon removal *

pushing

determinization *

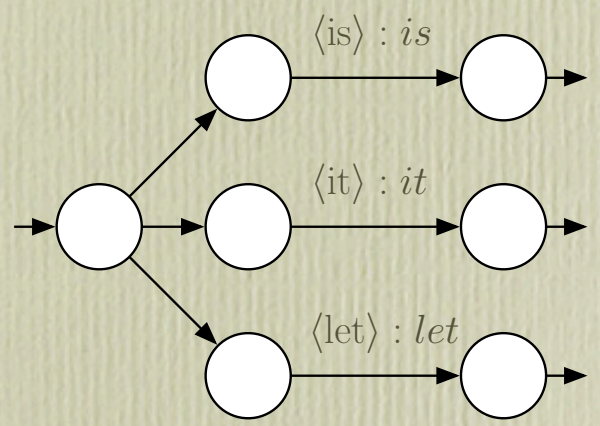
minimization *

inversion

composition

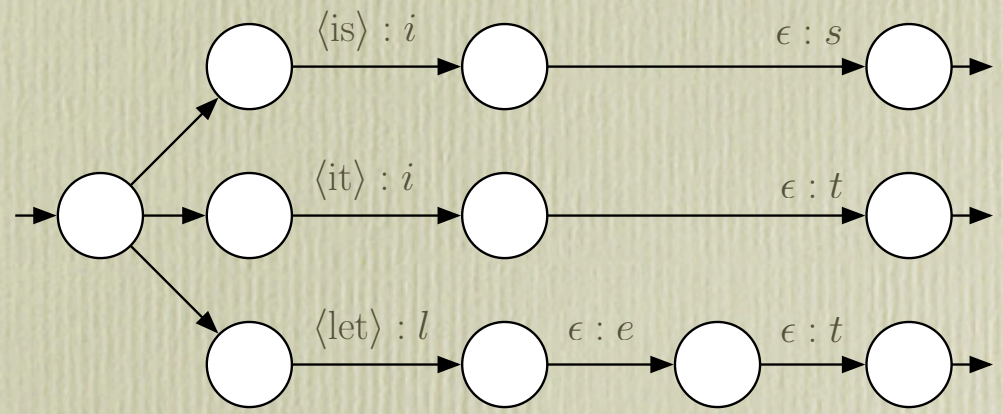
* sometimes

Granularity of Output

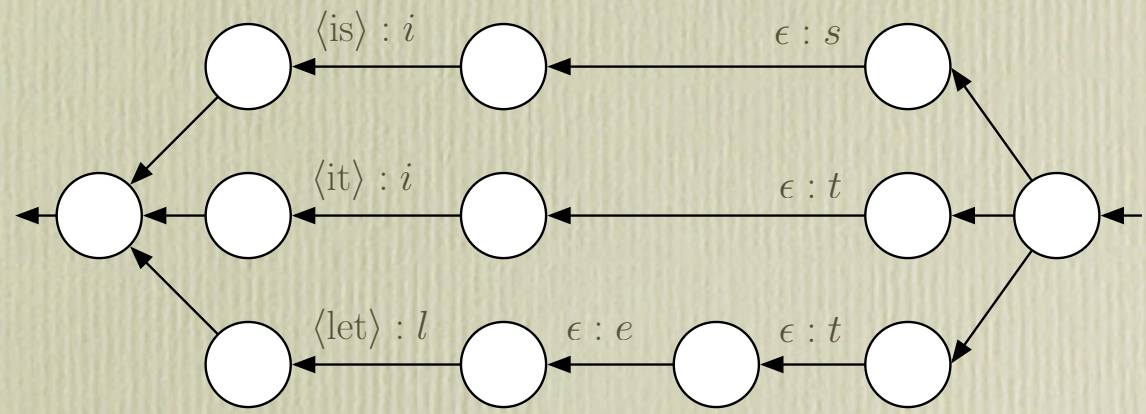


A Spelling Dictionary

Granularity of Output

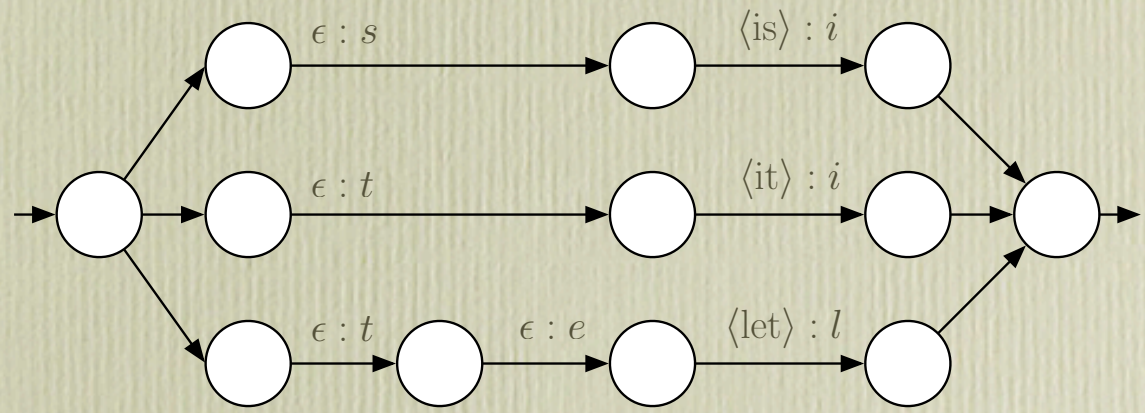


Left-Right Reversal



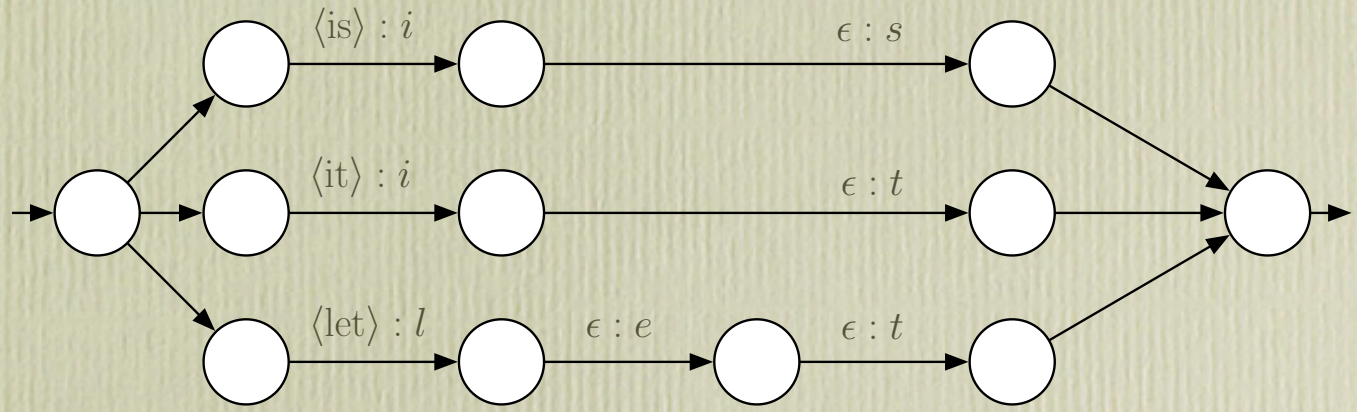
...by treating FST as FSA over cross-product vocabulary.

Left-Right Reversal

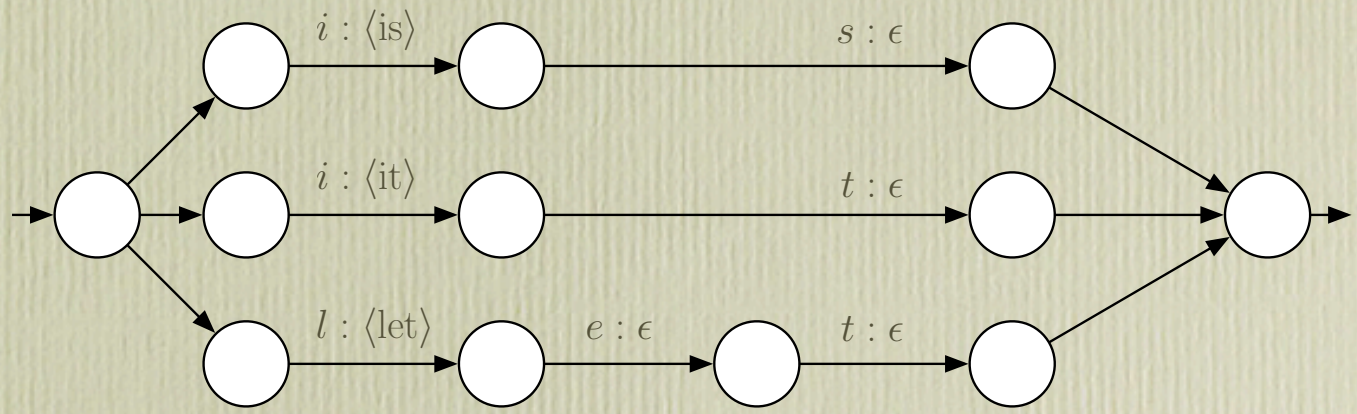


...by treating FST as FSA over cross-product vocabulary.

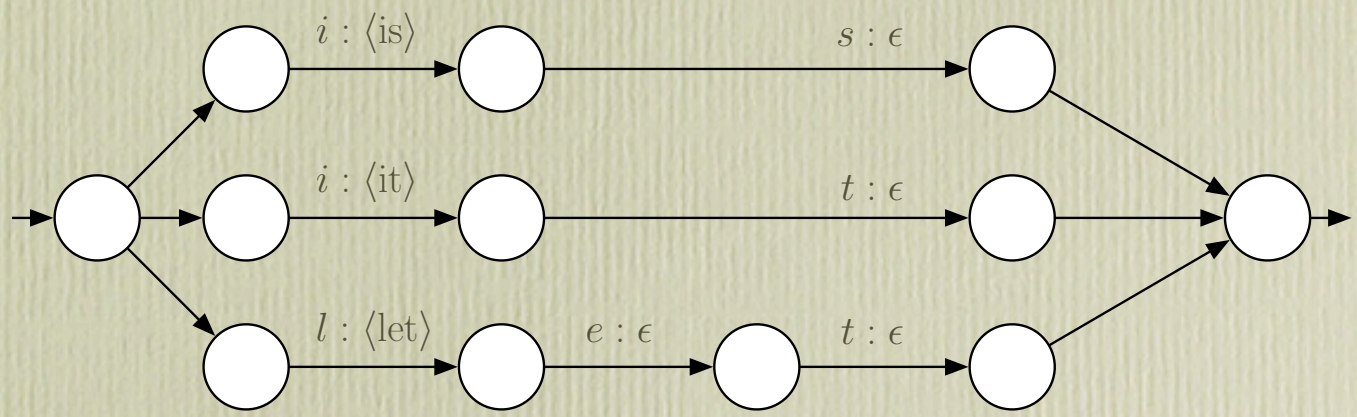
Inversion



Inversion

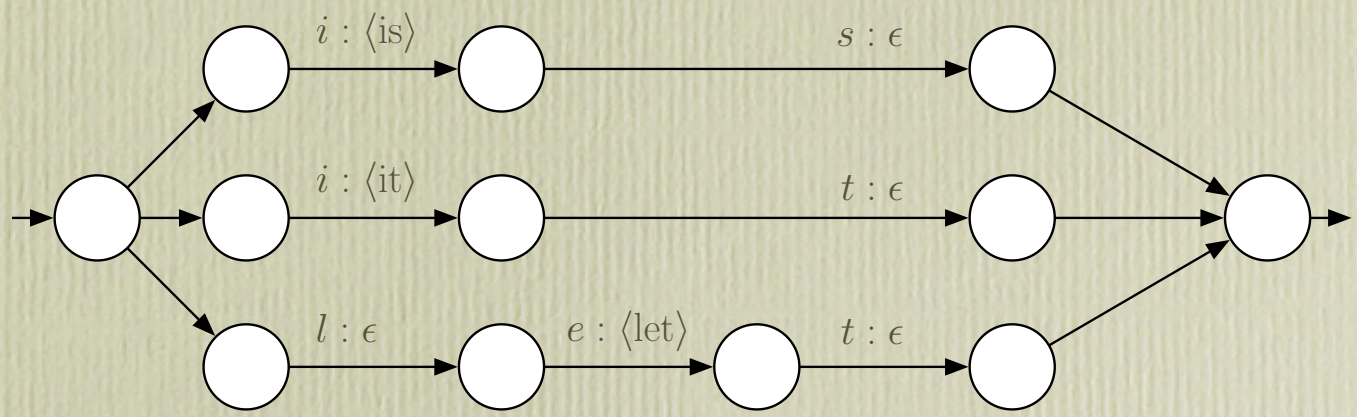

$i : \langle \text{is} \rangle$
$s : \epsilon$
$i : \langle \text{it} \rangle$
$t : \epsilon$
$l : \langle \text{let} \rangle$
$e : \epsilon$
$t : \epsilon$

Pushing

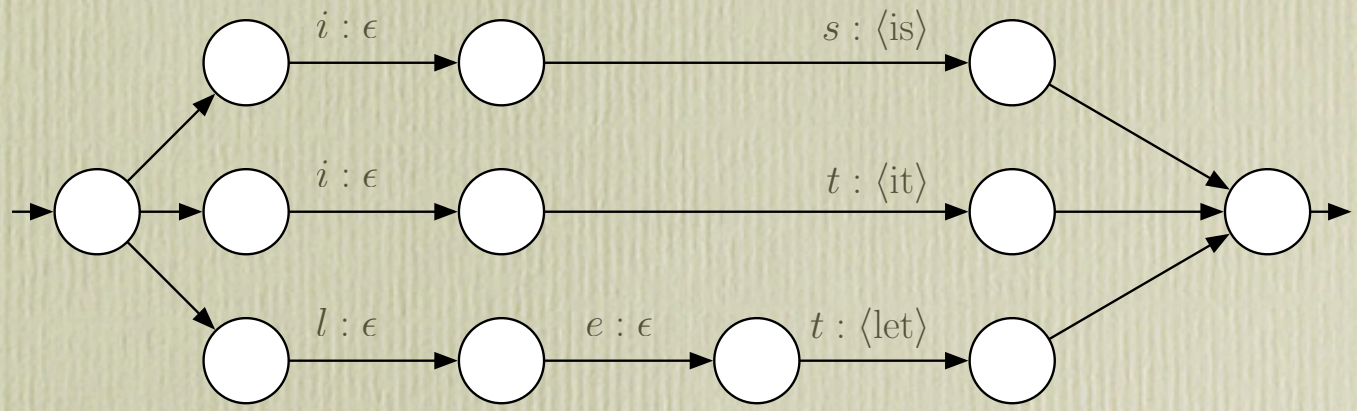


Movement of output symbols along a path.

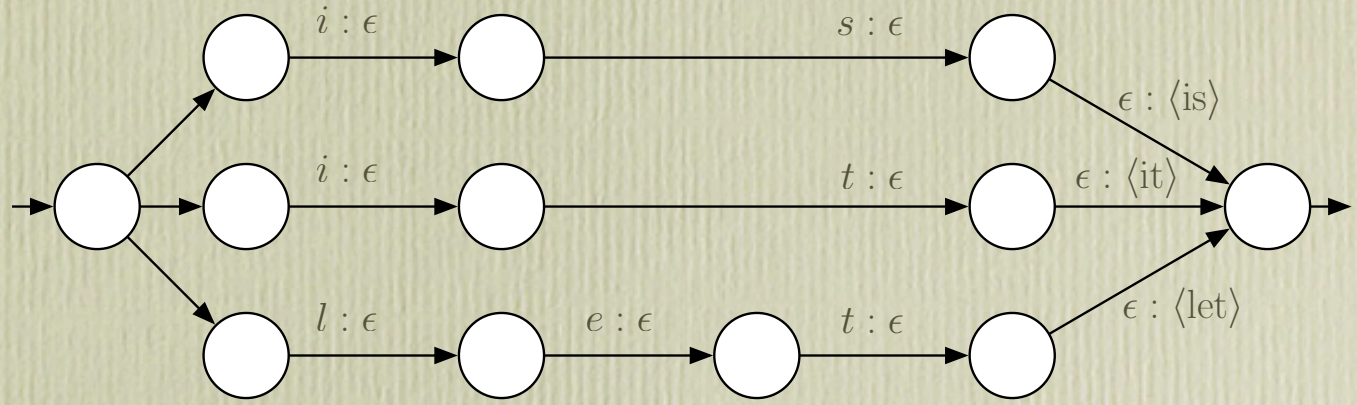
Pushing


$i : \langle \text{is} \rangle$
$s : \epsilon$
$i : \langle \text{it} \rangle$
$t : \epsilon$
$l : \epsilon$
$e : \langle \text{let} \rangle$
$t : \epsilon$

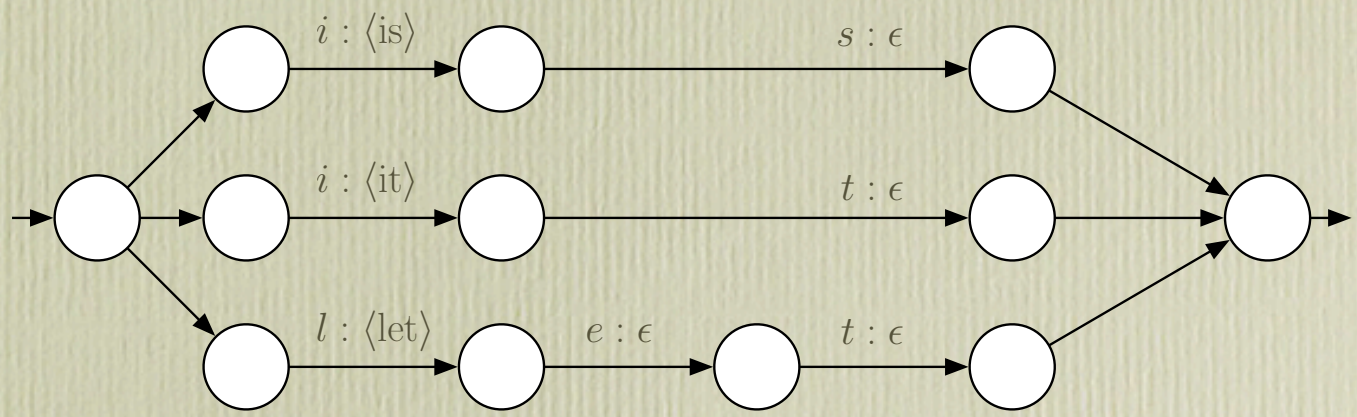
Pushing


$i : \epsilon$
$s : \langle \text{is} \rangle$
$i : \epsilon$
$t : \langle \text{it} \rangle$
$l : \epsilon$
$e : \epsilon$
$t : \langle \text{let} \rangle$

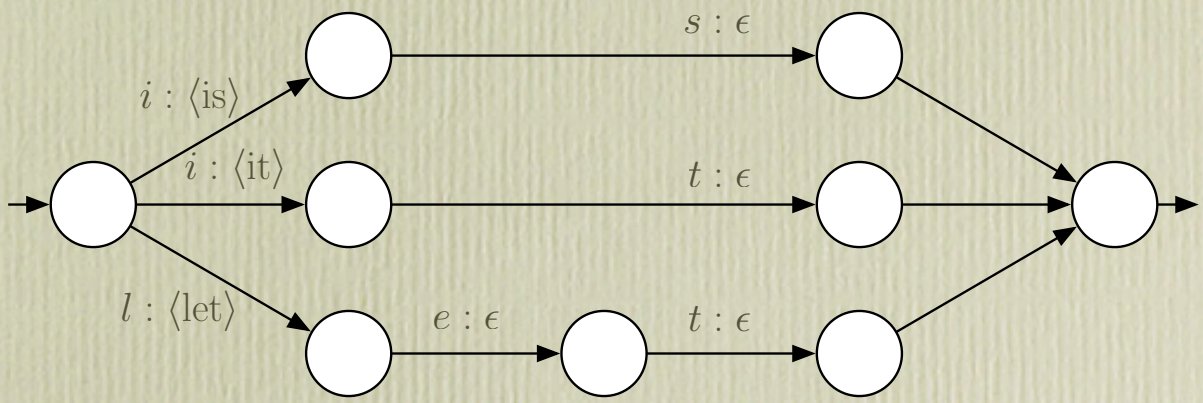
Pushing


$i : \epsilon$
$s : \epsilon$
$\epsilon : \langle \text{is} \rangle$
$i : \epsilon$
$t : \epsilon$
$\epsilon : \langle \text{it} \rangle$
$l : \epsilon$
$e : \epsilon$
$t : \epsilon$
$\epsilon : \langle \text{let} \rangle$

Determinization

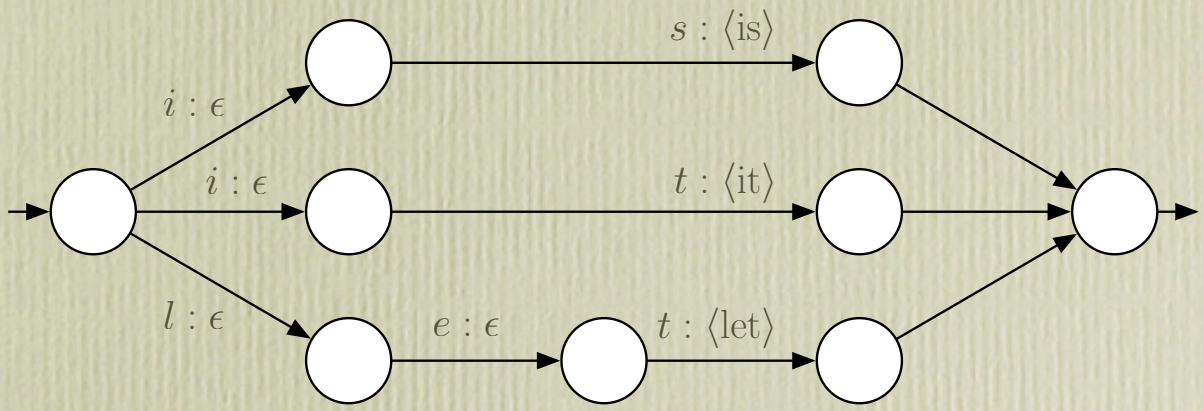


Determinization



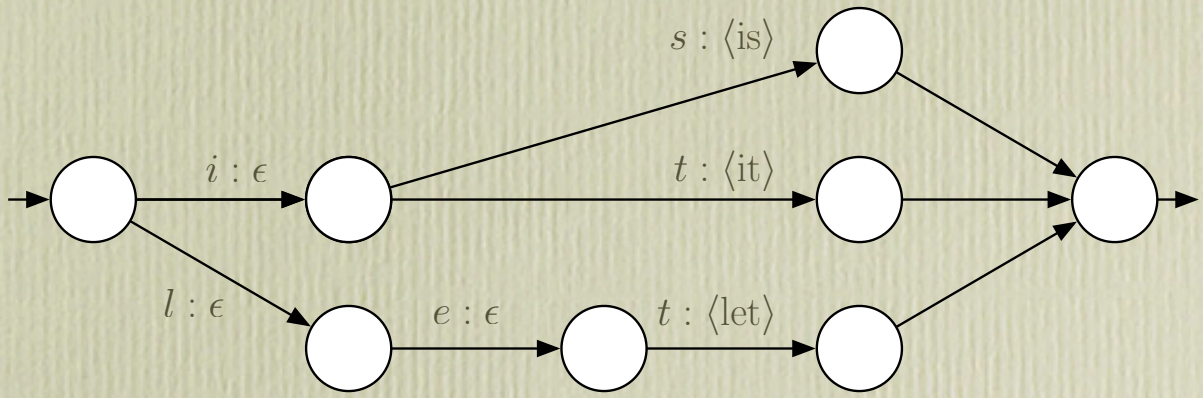
No more determinization possible...

Determinization



No determinization possible (without pushing)...

Determinization



(The transducer determinization algorithm performs forward pushing implicitly.)

Composition

Given two transducers *T* and *T'*, connect output of first to input of second

Result is a transducer *T* o *T'*

Application: Morphological Parser

Overview

‹sky› ‹+s› ‹wb› ‹child› ‹+s› ‹wb›

Language Model

‹sky› ‹+s› ‹wb› ‹child› ‹+s› ‹wb›

Irregular Spelling

‹sky› ‹+s› ‹wb› `children` ‹wb›

Regular Spelling

`sky+s_children_`

Orthography Model

`skies_children_`

Application: Morphological Parser

Overview

```
%% Language model: nouns with optional plural marker
%% separated by word boundaries
macro(lm, [nouns, option('<+s>'), '<wb>']*).

%% Nouns
macro(nouns, id({'<boy>',
     '<child>',
     '<sky>',
     '<box>'
     })).
```


Application: Morphological Parser

Overview

```
%% Replace irregular inflected forms with their spelling
macro(spellirreg,
      replace(['<child>', '<+s>'] x word(children), [], [])).

%% Replace regular forms with their spelling
macro(spellreg, {'<boy>':word('boy'),
     '<child>':word('child'),
     '<sky>':word('sky'),
     '<box>':word('box'),
     '<+s>':word('+s'),
     '<wb>':' ',
     id(a..z)
    }*).
```


Application: Morphological Parser

Overview

```
%% Consonants
macro(consonant, {b, c, d, f, g, h, j, k, l, m, n,
      p, q, r, s, t, v, w, x, y, z}).

%% Orthographic rules for pluralization
macro(ortho,
      replace(word('y+'):word('ie'), consonant, s)
       o replace('+':e, {s, z, x, word(ch), word(sh)}, s)
       o replace('+':[], [], s)
     ).

%% Morphological parser that inverts orthography
macro(parse, invert(lm o spellirreg o spellreg o ortho)).
```


Application: Morphological Parser

Demo

Weighting

Division of Engineering and Applied Sciences
Harvard University

Why Weights?

FSAs have multiple paths

How to adjudicate?

- Notion of *best path*

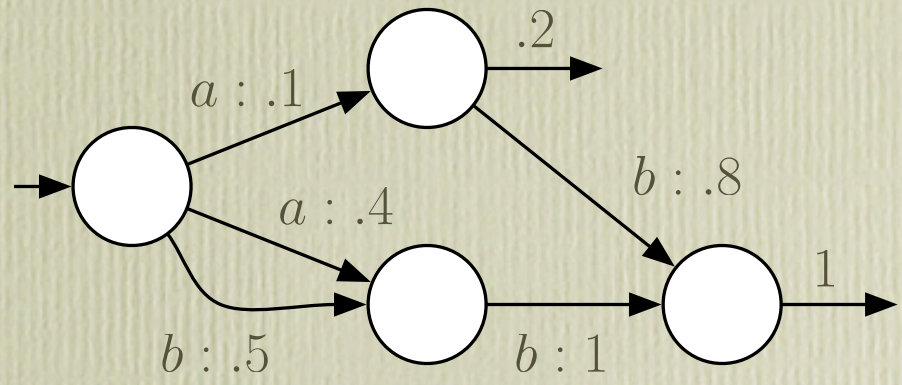
Other applications:

- Numeric functions over strings
  - perfect hashing

Weighted Finite-State Automaton

Defines probability distribution over strings:

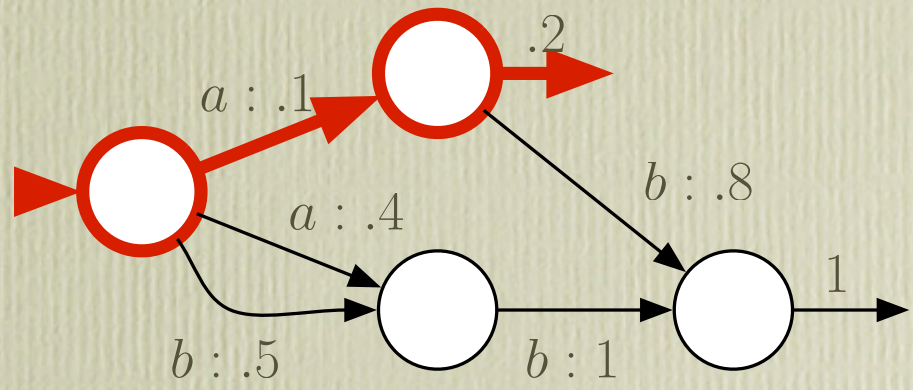
- *a*
- *bb*
- *ab*
- 1.0



Weighted Finite-State Automaton

Defines probability distribution over strings:

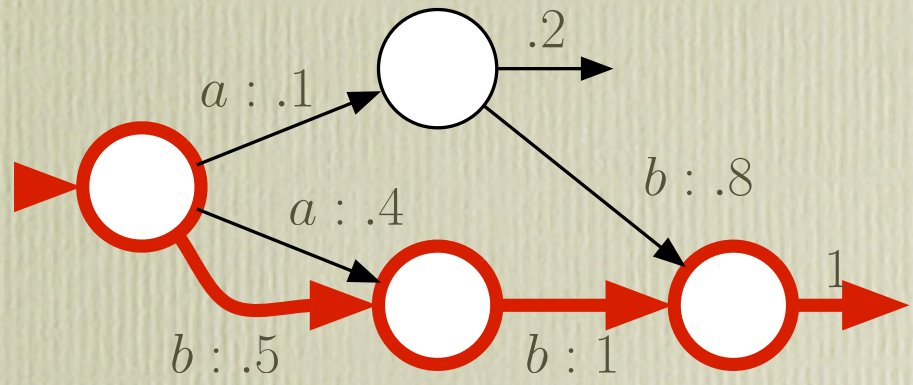
- *a* .02
- *bb*
- *ab*
- 1.0



Weighted Finite-State Automaton

Defines probability distribution over strings:

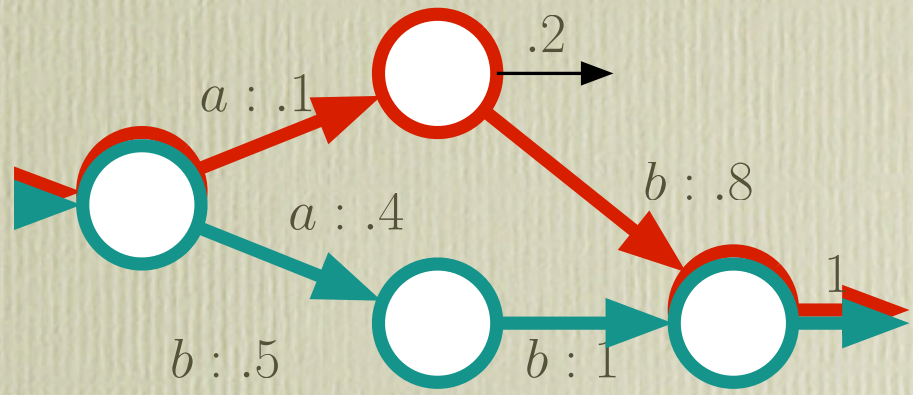
- *a* .02
- *bb* .5
- *ab*
- 1.0



Weighted Finite-State Automaton

Defines probability distribution over strings:

- *a* .02
- *bb* .5
- *ab* .48
- 1.0



Best Path

Best path through a weighted automaton

- Viterbi decoding
- Dijkstra's algorithm
- dynamic programming
  - computes score of best path from start state to each state

$$\delta_q(0) = \begin{cases} 1 & \text{if } q = q_0 \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_q(t+1) = \max_{\langle q', a:p, q\rangle \in \Delta} \delta_{q'}(t) \cdot p$$

  - for a given input, just intersect

Application:

Language Modeling

Goal: describe probability of strings of a language based on a sample (training corpus)

$$P(w_1 \cdots w_c) = \prod_{i=1}^{k} P(w_i \mid w_1 \cdots w_{i-1})$$

Approximate under a Markovian assumption that words depend only on the previous *N*–1

$$P(w_1 \cdots w_c) = \prod_{i=1}^{k} P(w_i \mid w_{i-N+1} \cdots w_{i-1})$$

Application: Language Modeling

Training

N-gram approximation:

$$P(w_1 \cdots w_c) = \prod_{i=1}^{k} P(w_i \mid w_{i-N+1} \cdots w_{i-1})$$

Maximum likelihood estimates of component N-gram probabilities:

$$P(w_i \mid w_{i-N+1} \cdots w_{i-1}) \approx \frac{c(w_{i-N+1} \cdots w_i)}{c(w_{i-N+1} \cdots w_{i-1})}$$

To start:

$$P(w_1 \mid w_{-N} \cdots w_0) = P(w_1 \mid \overbrace{\vdash \cdots \vdash}^{N-1 \text{ times}})$$

Let it be when it is mine to be sure let it be when it is mine when it is mine let it be to be sure when it is mine to be sure let it be let it be let it be to be sure let it be to be sure when it is mine to be sure let it to be sure when it is mine let it be to be sure let it be to be sure to be sure let it be to be sure let it be to be sure to be sure let it be to be sure let it be to be sure let it be to be sure let it be mine to be sure let it be to be sure to be mine to be sure to be mine to be sure to be mine let it be to be mine let it be to be sure to be mine to be sure let it be to be mine let it be to be sure let it be to be sure to be sure let it to be sure mine to be sure let it be mine to let it be to be sure to let it be mine when to be sure when to be sure to let it to be sure to be mine.

— Gertrude Stein,
An Acquaintance With Description, 1929

Application: Language Modeling

Example

Word	*Count*	*1-gram MLE*
be	62	.276
to	41	.182
it	33	.147
sure	31	.138
let	27	.120
mine	17	.076
when	8	.036
is	6	.027
total	225	

Let it be when it is mine to be sure let it be when it is mine when it is mine let it be to be sure when it is mine to be sure let it be let it be let it be to be sure let it be to be sure when it is mine to be sure let it to be sure when it is mine let it be to be sure let it be to be sure to be sure let it be to be sure let it be to be sure to be sure let it be to be sure let it be to be sure let it be mine to be sure let it be to be sure to be mine to be sure to be mine to be sure to be mine let it be to be mine let it be to be sure to be mine to be sure let it be to be mine let it be to be sure let it be to be sure to be sure let it to be sure mine to be sure let it be mine to let it be to be sure to let it be mine when to be sure when to be sure to let it to be sure to be mine.

— Gertrude Stein,
An Acquaintance With Description, 1929

Application: Language Modeling

Example

Trigram	*MLE Prob*
it be be	0
it be is	0
it be it	0
it be let	0.083
it be mine	0.125
it be sure	0
it be to	0.708
it be when	0.083

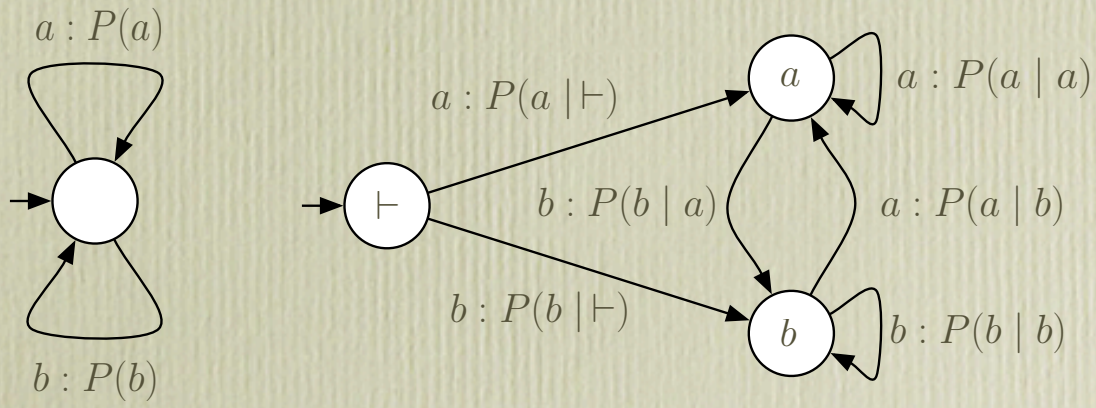
Application: Language Modeling

WFSA for N-gram Models

One state per (N–1-gram) conditioning context

Transitions among states to change context

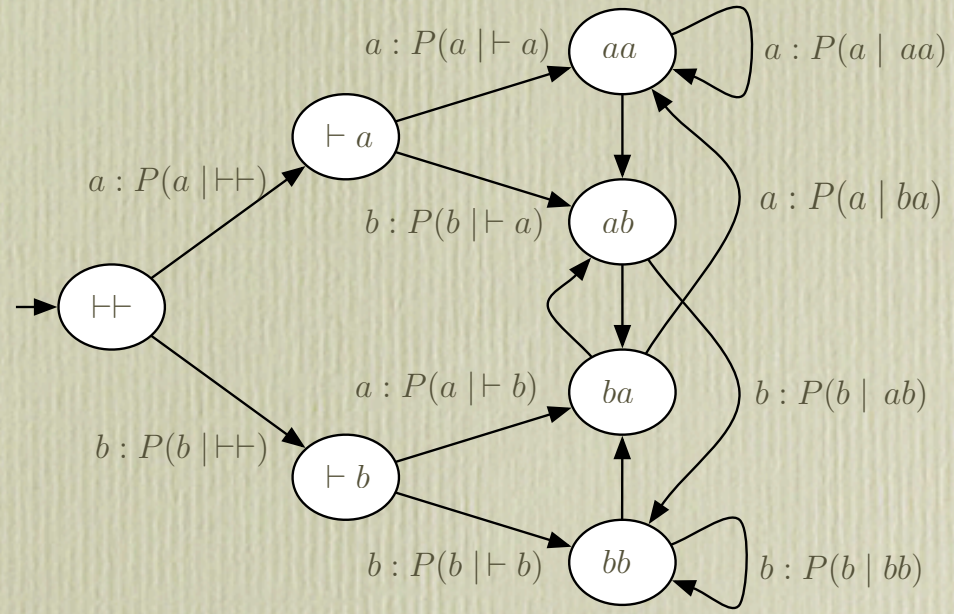
Start state is $\langle \vdash^{N-1} \rangle$



Unigram *Bigram*

Application: Language Modeling

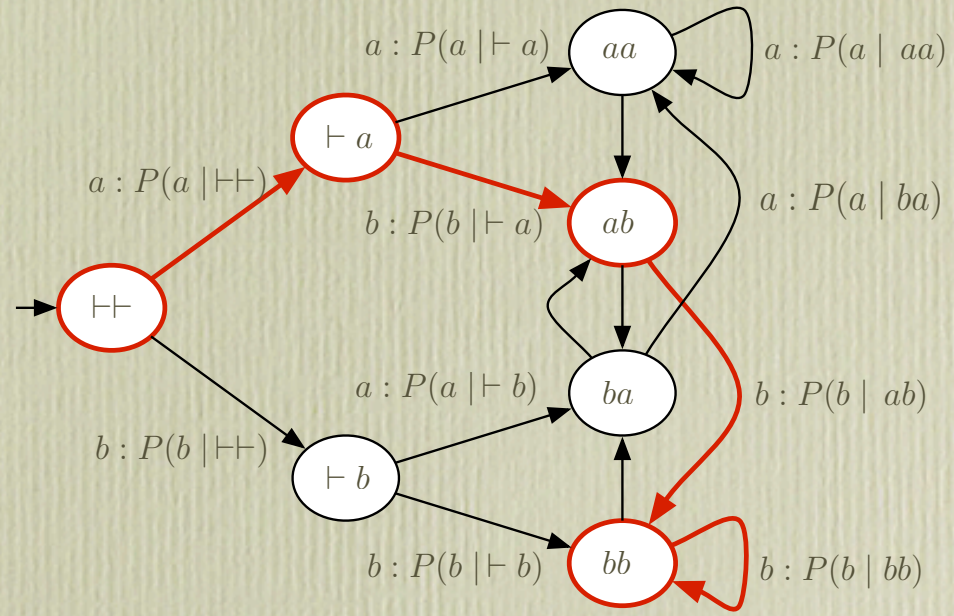
WFSA for N-gram Models



Trigram

Application: Language Modeling

WFSA for N-gram Models



$$P(abbb) = P(a \mid \vdash\vdash) \cdot P(b \mid \vdash a) \cdot P(b \mid ab) \cdot P(b \mid bb)$$

Weighted Transducers

Combining output and weighting

Examples:

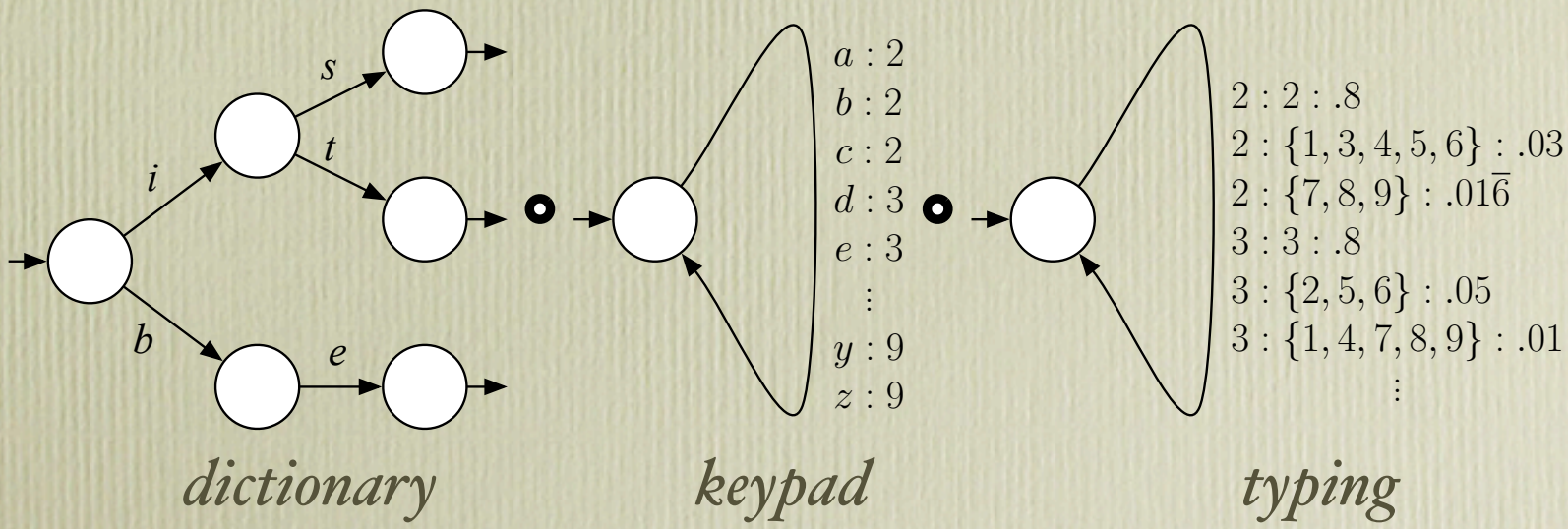
- typing models

Weighted Transducers

Combining output and weighting

Examples:

- typing models



Division of Engineering and Applied Sciences
Harvard University

Weighted Transducers

Combining output and weighting

Examples:

- typing models
- abbreviation models

  Drop all vowels after the first character
  Drop all but one repeated consonants

if y cn rd ths, y cn gt a gd jb

Abbreviation Decoding

‹an› ‹example› ‹of› ‹NUM› ‹words›

Language Model

‹an› ‹example› ‹of› ‹NUM› ‹words›

Spelling Model

an_example_of_‹NUM›_words

Compression Model

an_exmpl_of_‹NUM›_wrds

Unknowns Model

an_exmpl_of_5_wrds

Abbreviation Decoding

Demo

Original Text

[cnn.com; 387 characters; 60 words]

Chinese military officials have boarded a grounded U.S. surveillance plane and removed equipment from it despite U.S. protests. In a signal that a standoff between the two nations is not likely to end soon, Pentagon sources told CNN today that China had begun removing sensitive eavesdropping equipment from the plane. Meanwhile, U.S. and Chinese diplomats were meeting on Hainan Island.

Abbreviation Decoding

Demo

Abbreviated Text

[279 characters]

Chns mltry ofcls hv brdd a grndd U.S. srvlnc pln and rmvd eqpmnt frm it dspt U.S. prtsts. In a sgnl tht a stndf btwn th tw ntns is nt lkly t end sn, Pntgn srcs tld CNN tdy tht Chn hd bgn rmvng snstv evsdrpng eqpmnt frm th pln. Mnwhl, U.S. and Chns dplmts wr mtng on Hainan Islnd.

Abbreviation Decoding

Demo

Reconstructed Text

[4 errors]

Chinese military officials have bearded a grounded U.S. surveillance plan and removed equipment from it despite U.S. protests. In a signal that a standoff between the two nations is not likely to end soon, Pentagon sources told CANON today that China had begun removing sensitive eavesdropping equipment from the plan. Meanwhile, U.S. and Chinese diplomats were meeting on Hainan Island.

Abbreviation Decoding

Demo

Original

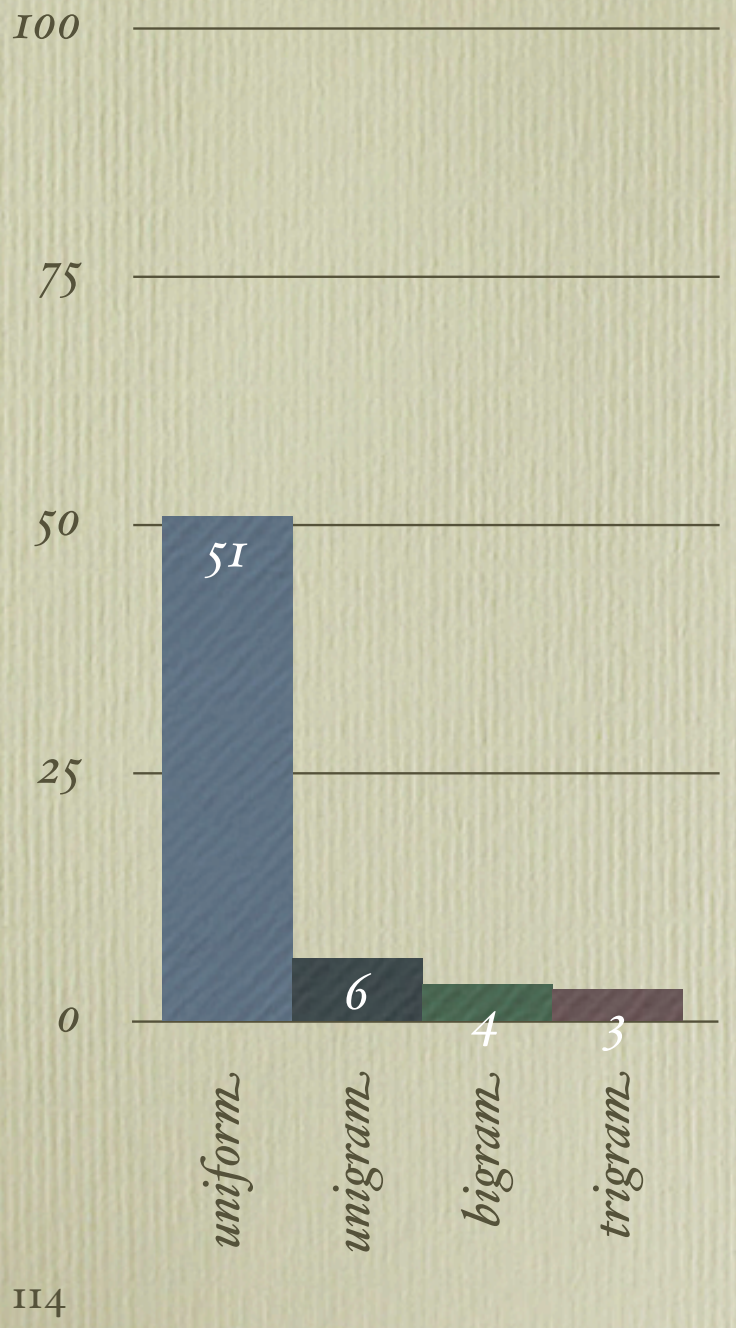
T mny ppl hv cm *t* th lctr *t* ft in th rm.

Reconstructed

Too many people have come *to* the lecture *to* fit in the room.

Abbreviation Decoding

Performance



Division of Engineering and Applied Sciences
Harvard University

Summary

Weighted finite-state transducers

- provide an elegant, uniform, formalism
- cover a vast range of low-level natural-language processing tasks
  - characterizable as string to string transformations

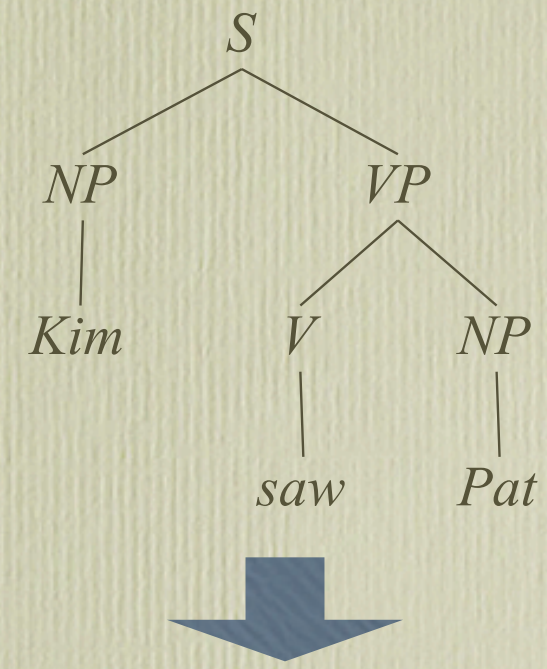
Generality based on properties such as

- composability (closure under composition)
- efficiency (determinizability and minimizability, enabled by pushing)
- weighting (for choice)

Tree Automata

Division of Engineering and Applied Sciences
Harvard University

Trees as Terms



$$S(NP(Kim),\ VP(V(saw), NP(Pat)))$$

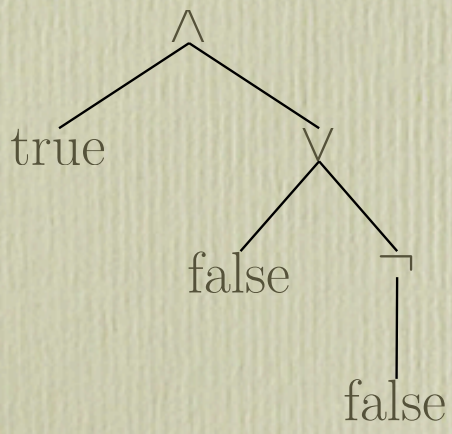
Example:

Propositional Formulae

Propositional formulae: $\mathcal{T}(\mathcal{F}_{prop})$

$$\mathcal{F}_{prop} = \{\wedge_2, \vee_2, \neg_1, \text{TRUE}_0, \text{FALSE}_0\}$$

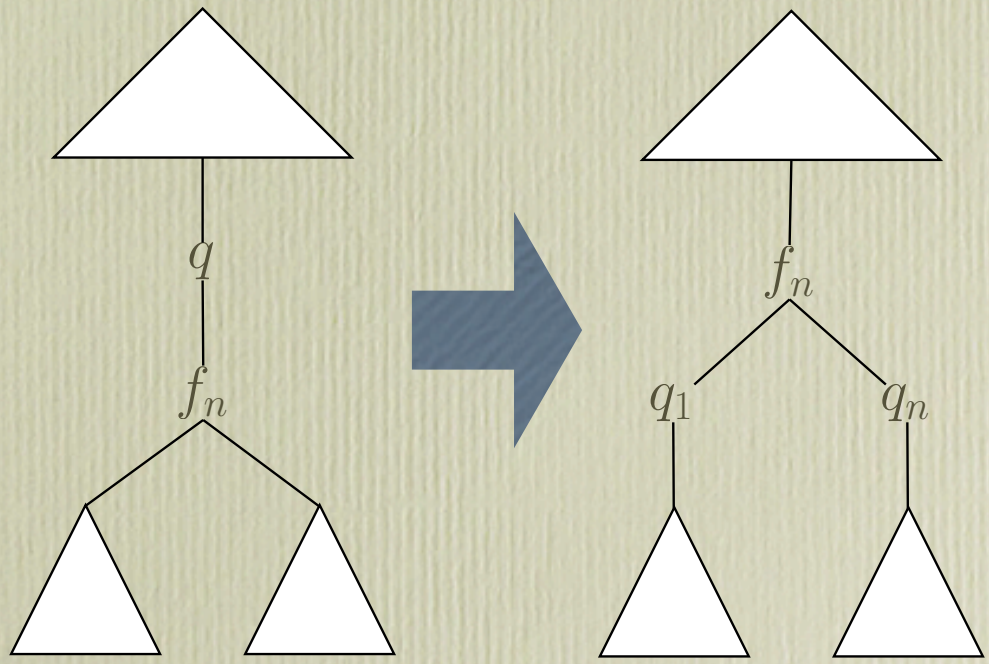
(Arities are given in the subscripts.)



$\text{true} \wedge (\text{false} \vee \neg\text{false})$

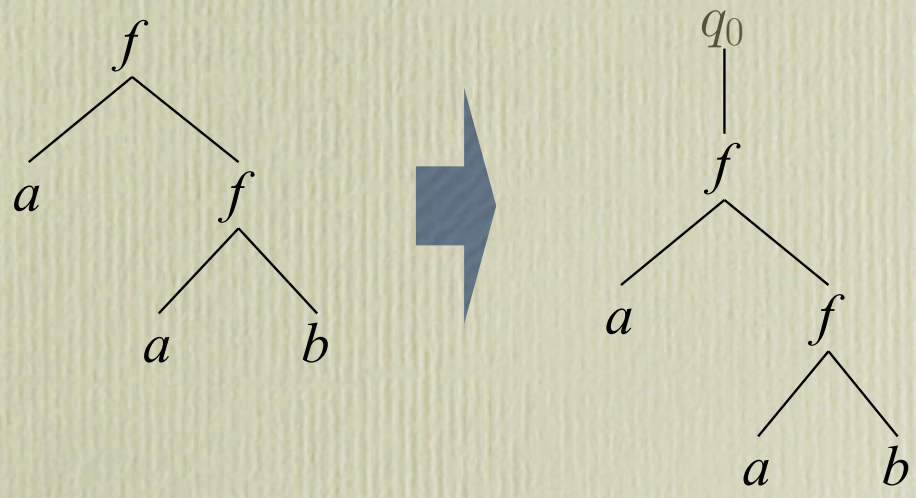
Tree Automaton Transitions

$$q(f_n(x_1, \ldots, x_n)) \rightarrow f_n(q_1(x_1), \ldots, q_n(x_n))$$



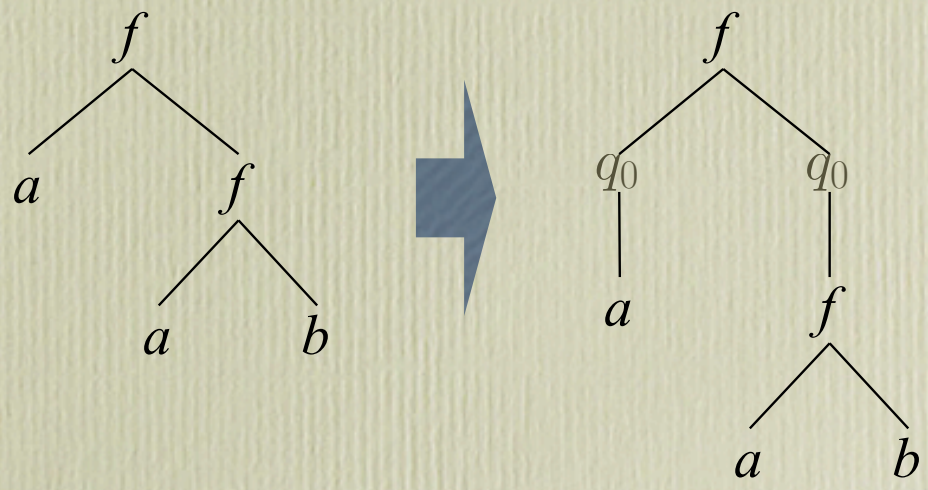
Example Tree Automaton

- $q_0(f(x_1, x_2)) \rightarrow f(q_0(x_1), q_0(x_2))$
  $q_0(a) \rightarrow a$
  $q_0(b) \rightarrow b$



Example Tree Automaton

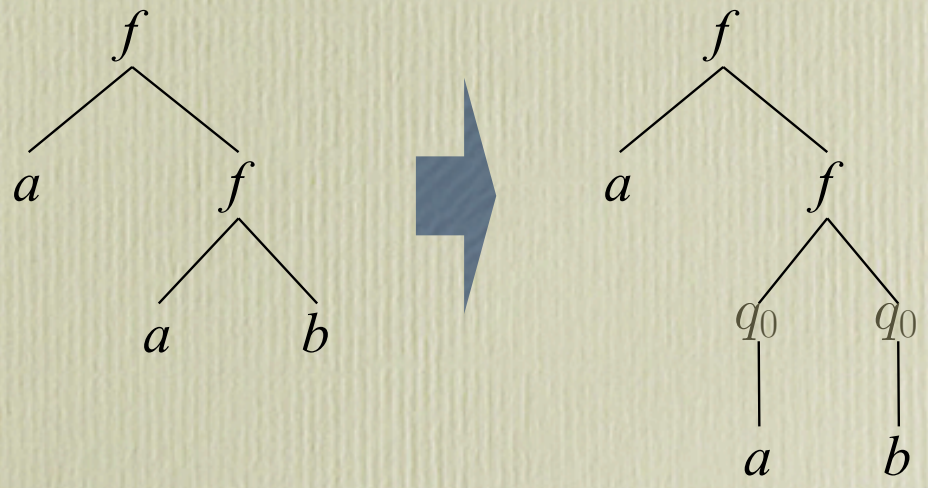
- $q_0(f(x_1, x_2)) \rightarrow f(q_0(x_1), q_0(x_2))$
- $q_0(a) \rightarrow a$
  $q_0(b) \rightarrow b$



Example Tree Automaton

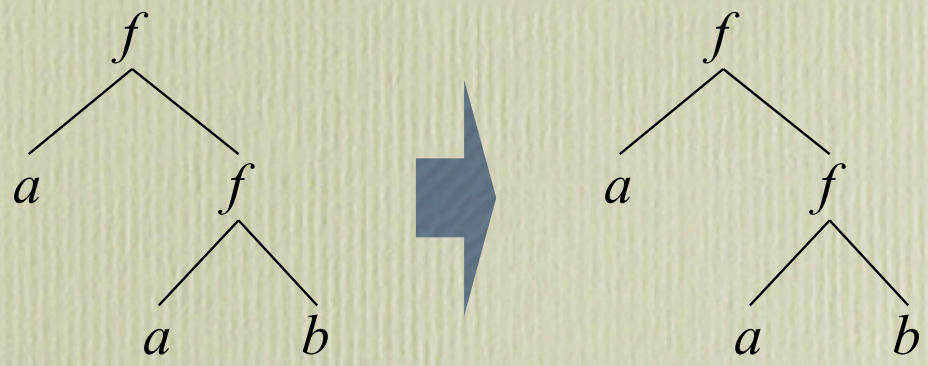
$q_0(f(x_1, x_2)) \rightarrow f(q_0(x_1), q_0(x_2))$

- $q_0(a) \rightarrow a$
- $q_0(b) \rightarrow b$



Example Tree Automaton

$q_0(f(x_1, x_2)) \rightarrow f(q_0(x_1), q_0(x_2))$
$q_0(a) \rightarrow a$
$q_0(b) \rightarrow b$



Example:

Recognizing Valid Formulae

$$
\begin{aligned}
q_t(\text{TRUE}) &\rightarrow \text{TRUE} \\
q_f(\text{FALSE}) &\rightarrow \text{FALSE}
\end{aligned}
$$

$$
\begin{aligned}
q_t(\neg x_1) &\rightarrow \neg q_f(x_1) \\
q_f(\neg x_1) &\rightarrow \neg q_t(x_1)
\end{aligned}
$$

$$
\begin{aligned}
q_t(x_1 \wedge x_2) &\rightarrow q_t(x_1) \wedge q_t(x_2) \\
q_f(x_1 \wedge x_2) &\rightarrow q_t(x_1) \wedge q_f(x_2) \\
q_f(x_1 \wedge x_2) &\rightarrow q_f(x_1) \wedge q_t(x_2) \\
q_f(x_1 \wedge x_2) &\rightarrow q_f(x_1) \wedge q_f(x_2)
\end{aligned}
$$

$$
\begin{aligned}
q_t(x_1 \vee x_2) &\rightarrow q_t(x_1) \vee q_t(x_2) \\
q_t(x_1 \vee x_2) &\rightarrow q_t(x_1) \vee q_f(x_2) \\
q_t(x_1 \vee x_2) &\rightarrow q_f(x_1) \vee q_t(x_2) \\
q_f(x_1 \vee x_2) &\rightarrow q_f(x_1) \vee q_f(x_2)
\end{aligned}
$$

$$
\begin{aligned}
& q_t(\text{FALSE} \vee \neg\text{FALSE}) \\
& \rightarrow q_f(\text{FALSE}) \vee q_t(\neg\text{FALSE}) \\
& \rightarrow \text{FALSE} \vee q_t(\neg\text{FALSE}) \\
& \rightarrow \text{FALSE} \vee \neg q_f(\text{FALSE}) \\
& \rightarrow \text{FALSE} \vee \neg\text{FALSE}
\end{aligned}
$$

Division of Engineering and Applied Sciences
Harvard University

Properties of Tree Automata

Characterize context-free languages

Closure under

- (left-right reversal)
- top-down–bottom-up reversal
- union
- substitution
- iterative substitution
- granularity
- epsilon removal: $q(x) \rightarrow q'(x)$
- determinization: bottom-up only

Tree Transducers

Division of Engineering and Applied Sciences
Harvard University

Tree Transducers

Regarding tree transformations, results do not flow so easily. Several definitions are candidate for the label 'tree transductions', with [unrelated] properties. People will keep in mind how gracefully behaved rational (word) transductions [are].

— Raoult, 1992

Generalizing Transitions

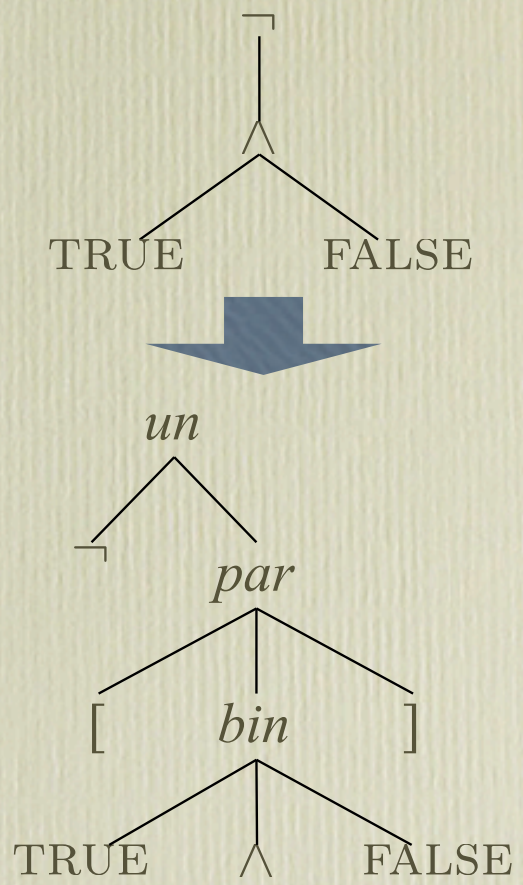
$$q(f_n(x_1, \ldots, x_n)) \rightarrow f_n(q_1(x_1), \ldots, q_n(x_n))$$

$$q(f_n(x_1, \ldots, x_n)) \rightarrow T[q_1(x_1), \ldots, q_n(x_n)]$$

Example:

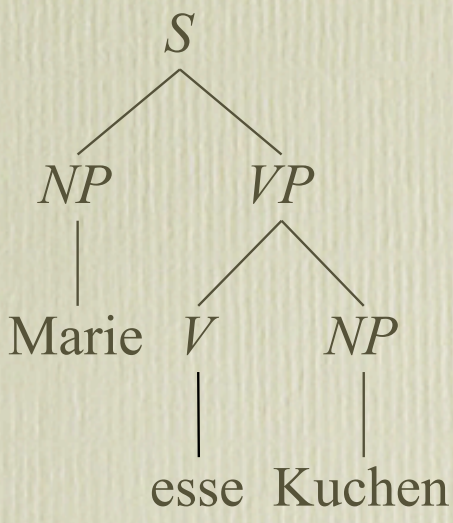
Concrete Syntax of Formulae

$$
\begin{array}{ll}
q_0(\neg x) & \rightarrow un(\neg, q_{\wedge\vee}(x)) \\
q_0(x \wedge y) & \rightarrow bin(q_{\wedge\vee}(x), \wedge, q_{\vee}(y)) \\
q_0(x \vee y) & \rightarrow bin(q_{\vee}(x), \vee, q_0(y)) \\
q_0(\text{true}) & \rightarrow \text{true} \\
q_0(\text{false}) & \rightarrow \text{false} \\
q_{\wedge\vee}(\neg x) & \rightarrow un(\neg, q_{\wedge\vee}(x)) \\
q_{\wedge\vee}(x \wedge y) & \rightarrow par([, bin(q_{\wedge\vee}(x), \wedge, q_{\vee}(y)),]) \\
q_{\wedge\vee}(x \vee y) & \rightarrow par([, bin(q_{\vee}(x), \vee, q_0(y)),]) \\
q_{\wedge\vee}(\text{true}) & \rightarrow \text{true} \\
q_{\wedge\vee}(\text{false}) & \rightarrow \text{false} \\
q_{\vee}(\neg x) & \rightarrow un(\neg, q_{\wedge\vee}(x)) \\
q_{\vee}(x \wedge y) & \rightarrow bin(q_{\wedge\vee}(x), \wedge, q_{\vee}(y)) \\
q_{\vee}(x \vee y) & \rightarrow par([, bin(q_{\vee}(x), \vee, q_0(y)),]) \\
q_{\vee}(\text{true}) & \rightarrow \text{true} \\
q_{\vee}(\text{false}) & \rightarrow \text{false}
\end{array}
$$



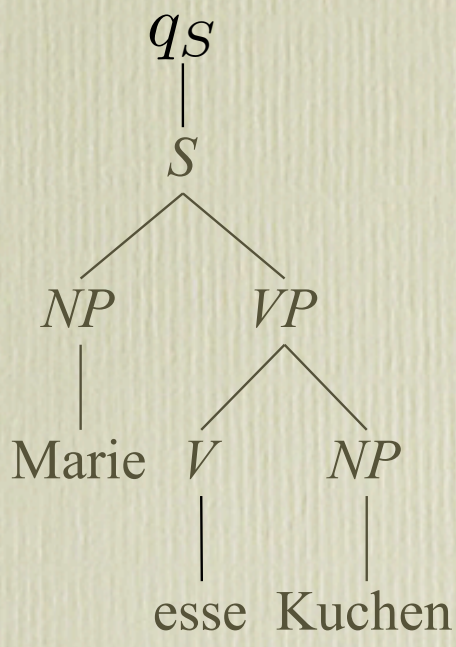
Example MT Transfer

$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



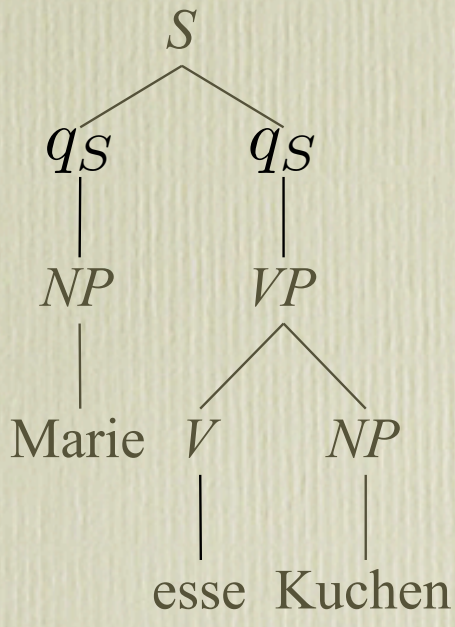
Example MT Transfer

$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



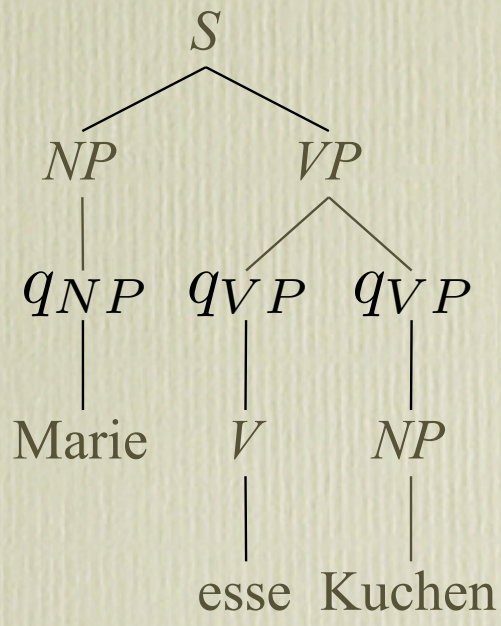
Example MT Transfer

$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



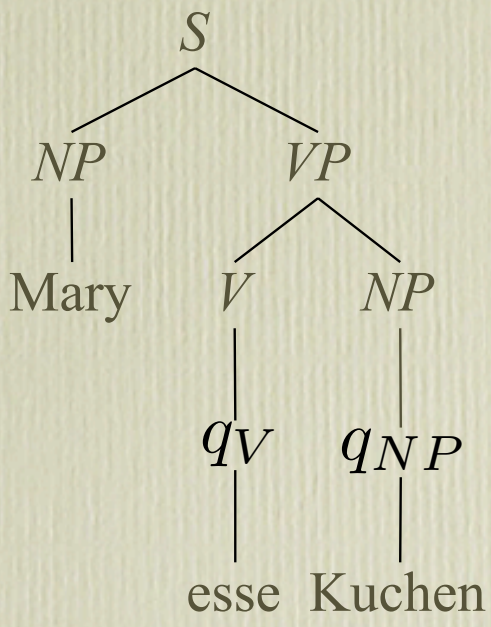
Example MT Transfer

$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



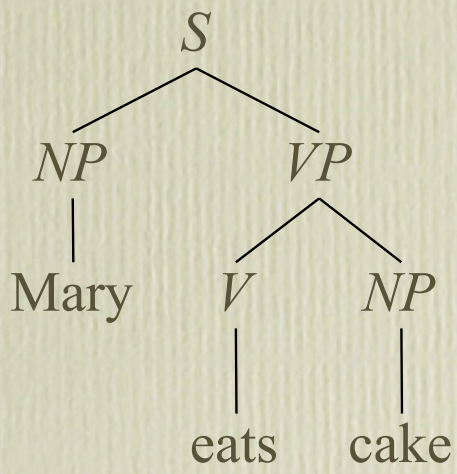
Example MT Transfer

$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



Example MT Transfer

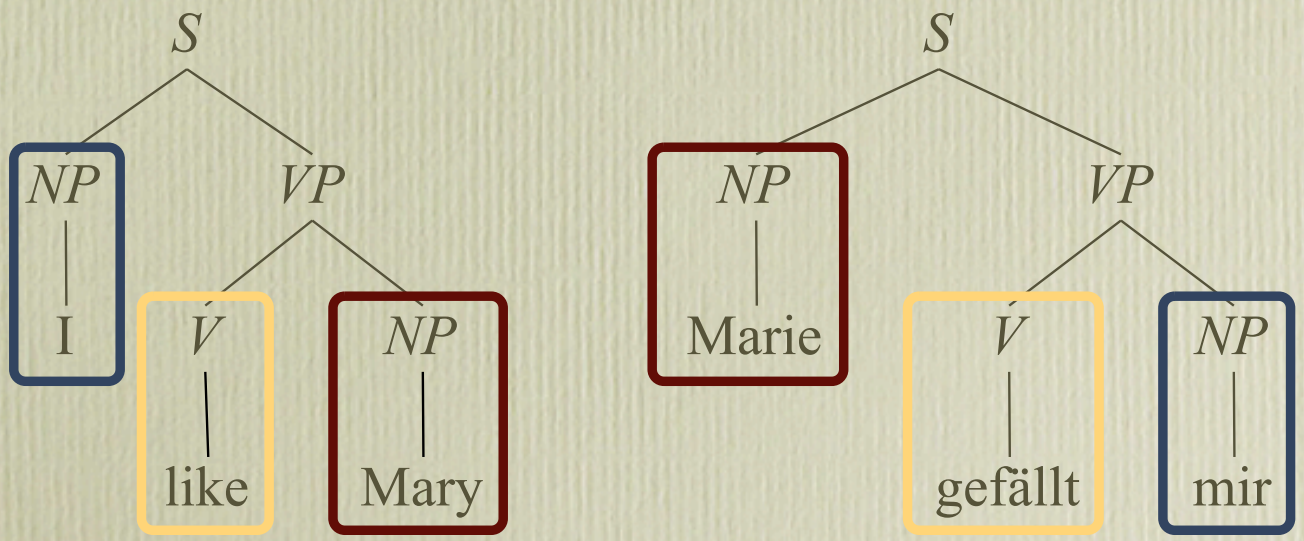
$$
\begin{aligned}
q_S(S(x_1, x_2)) &\rightarrow S(q_S(x_1), q_S(x_2)) \\
q_S(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_S(VP(x_1, x_2)) &\rightarrow VP(q_{VP}(x_1), q_{VP}(x_2)) \\
q_{VP}(V(x_1)) &\rightarrow V(q_V(x_1)) \\
q_{VP}(NP(x_1)) &\rightarrow NP(q_{NP}(x_1)) \\
q_{NP}(\text{Marie}) &\rightarrow \text{Mary} \\
q_{NP}(\text{Kuchen}) &\rightarrow \text{cake} \\
q_V(\text{esse}) &\rightarrow \text{eats}
\end{aligned}
$$



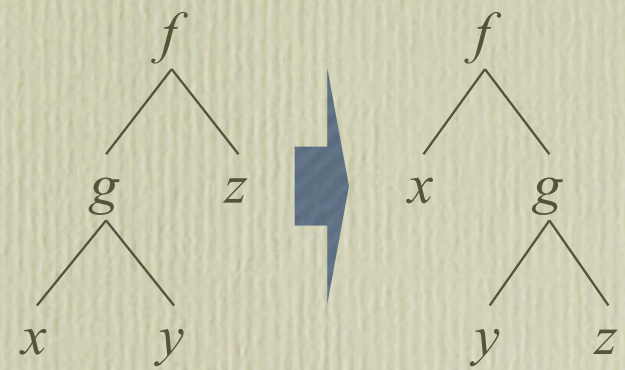
Why Rotations?

Consider

- I like Mary.
- Marie gefällt mir.



Expressing Rotations



$$q(f(g(x,y),z)) \rightarrow f(q(x),g(q(y),q(z)))$$

$$\begin{aligned} q(f(x_{gxy},z)) &\rightarrow f(q_1(x_{gxy}),g(q_2(x_{gxy}),q(z))) \\ q_1(g(x,y)) &\rightarrow q(x) \\ q_2(g(x,y)) &\rightarrow q(y) \end{aligned}$$

requires nonlinearity

Nonlinearity Generates Exponential Transductions

Generating perfect binary trees:

$$\begin{aligned} q(f(x)) &\to g(q(x), q(x)) \\ q(a) &\to a \end{aligned}$$

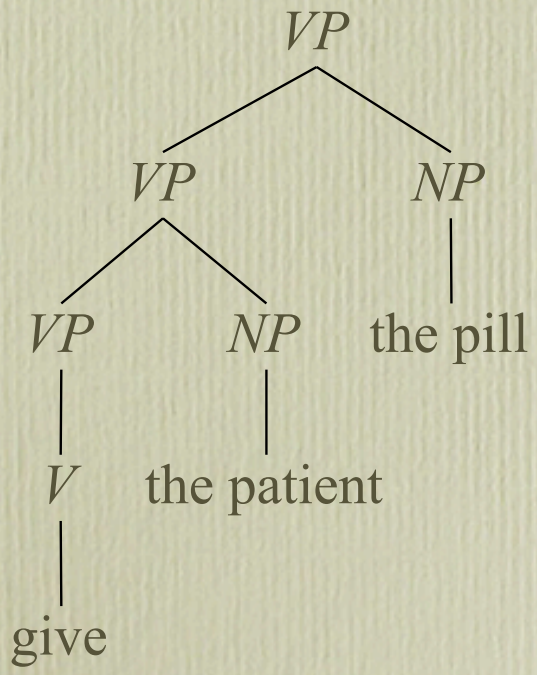
$$q(f(a)) \to g(q(a), q(a)) \to^* g(a, a)$$
$$q(f(f(a))) \to g(q(f(a)), q(f(a))) \to^* g(g(a, a), g(a, a))$$
$$q(f(f(f(a)))) \to^* g(g(g(a, a), g(a, a)), g(g(a, a), g(a, a)))$$
$$|q(f^n(a))| = 2^n - 1$$

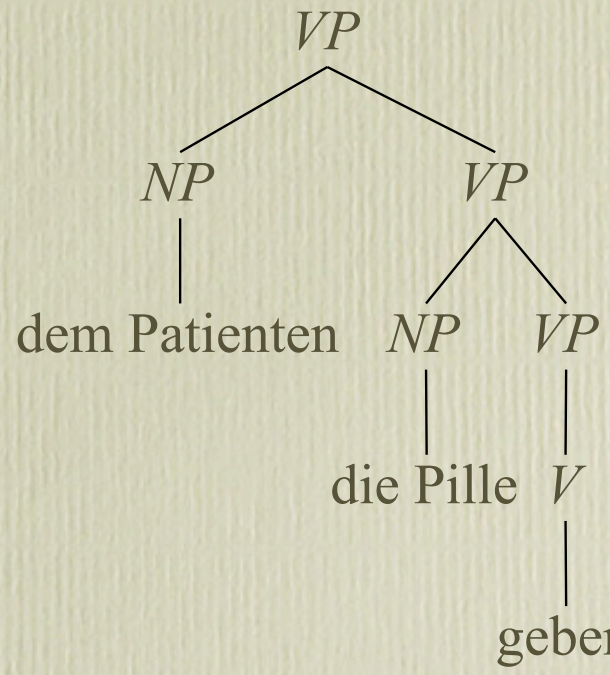
Exponential growth implies no composition closure

Why Global Rotations?

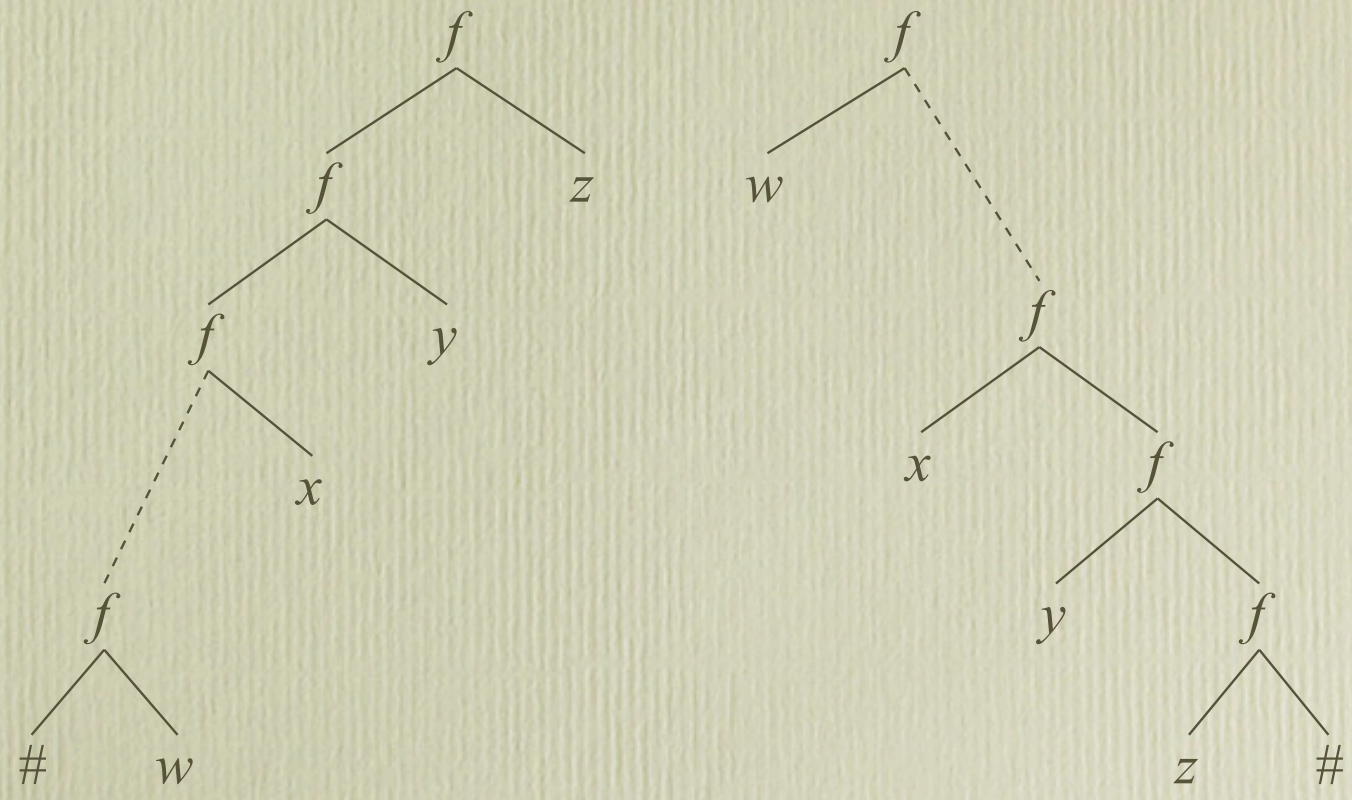
Consider

- Dann wird der Doktor dem patienten die Pille geben
- Then the doctor will give the patient the pill





No Global Rotations



cf. macro tree transducers

Summary

Linearity would be helpful

- closure under composition
- no exponential growth
- invertibility

But linearity is insufficient...

- no local rotation

and even nonlinear transducers are insufficient

- global rotations
- fringe

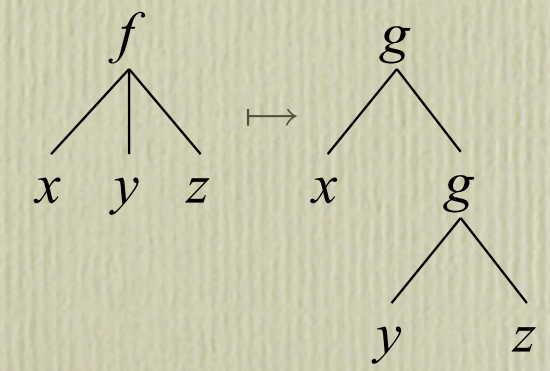
Extended Transducers: Bimorphisms

Division of Engineering and Applied Sciences
Harvard University

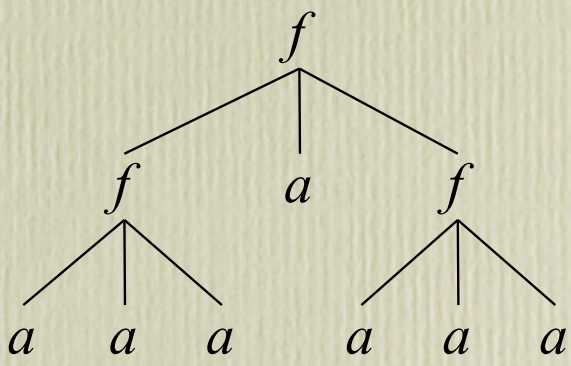
Tree Homomorphisms

Replace each symbol with a fixed subtree

Equivalent to one-state tree transducers



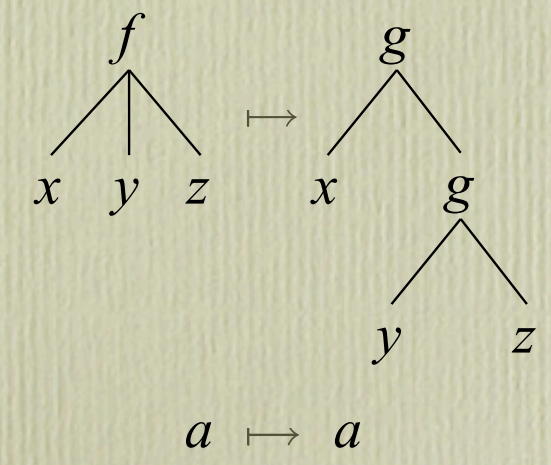
$a \longmapsto a$

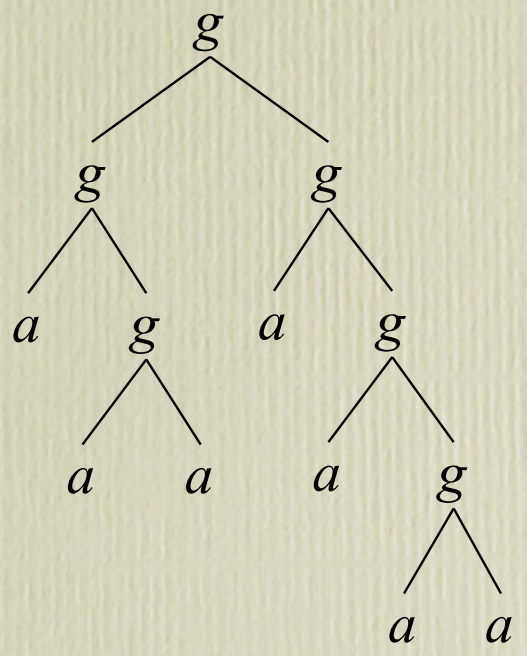


Tree Homomorphisms

Replace each symbol with a fixed subtree

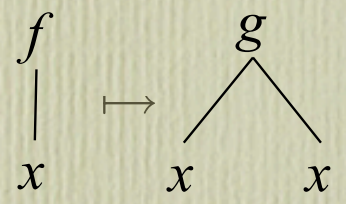
Equivalent to one-state tree transducers

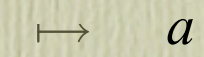


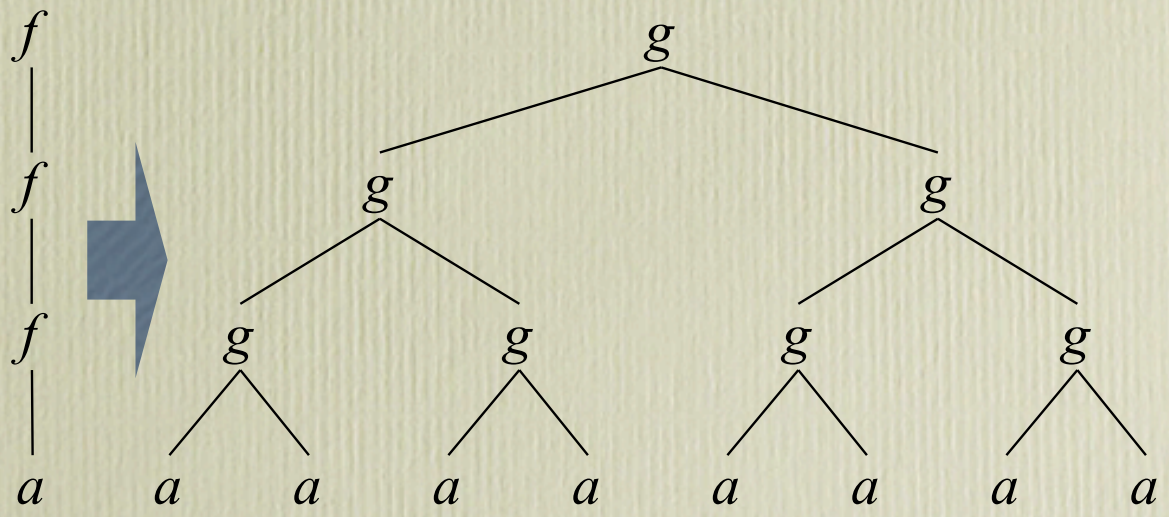


Example

Perfect Binary Trees







Restricting Homomorphisms

Linear (*L*)

- no repeated variables

Complete (*C*)

- no dropped variables

Epsilon-free (*F*)

- some structure on output

Symbol-to-symbol (*S*)

- output of height 1
- (implies epsilon-free)

Delabeling (*D*)

- = linear complete symbol-to-symbol

Bimorphisms

Bimorphism: $\langle h_{in}, L, h_{out} \rangle$

- regular (tree) language
- two homomorphisms

Bimorphisms define tree relations:

- $h_{in}^{-1} \circ L \circ h_{out}$

Intuition for Restriction

$$q(f_n(x_1, \ldots, x_n)) \to T[q_1(x_1), \ldots, q_n(x_n)]$$

$f_n(x_1, \ldots, x_n)$

$(\approx f_n)$

delabeling (*D*)

$T[x_1, \ldots, x_n]$

arbitrary morphism (*M*)

Bimorphism Characterization

The class of bottom-up tree transductions is equivalent to the relations defined by tree bimorphisms where the first homomorphism is a delabeling.

If the homomorphisms are linear (epsilon-free, complete), the bimorphism characterizes a linear (resp., epsilon-free, complete) transduction.

This asymmetry explains, e.g., lack of invertibility.

Types of Bimorphisms

$B(x, y)$ bimorphisms with homomorphisms of type x and y

- M any homomorphism
- L linear
- C complete
- F epsilon-free
- D delabeling

$B(D, M)$ equivalent to tree transducers

Regaining Symmetry

B(*M*, *M*)

- very powerful; composition is Turing-equivalent

B(*L*, *L*)

- expands input power; contracts output power
- not closed under composition
- $B(L, L) \subset B(L, L)^2 \subset B(L, L)^3 \subset B(L, L)^4 = B(L, L)^5$

B(*LCF*, *LCF*)

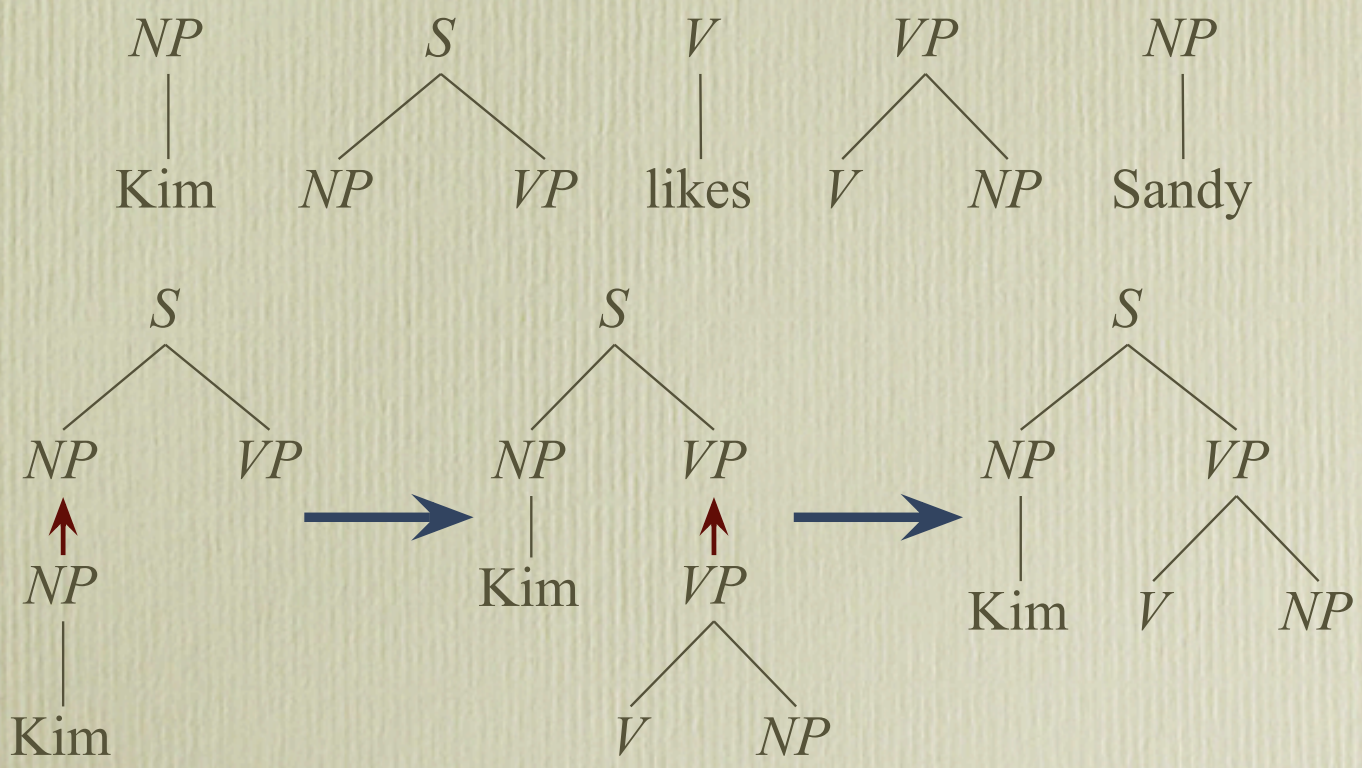
- $B(LCF, LCF) \subset B(LCF, LCF)^2 = B(LCF, LCF)^3$

B(*LC*, *LC*)

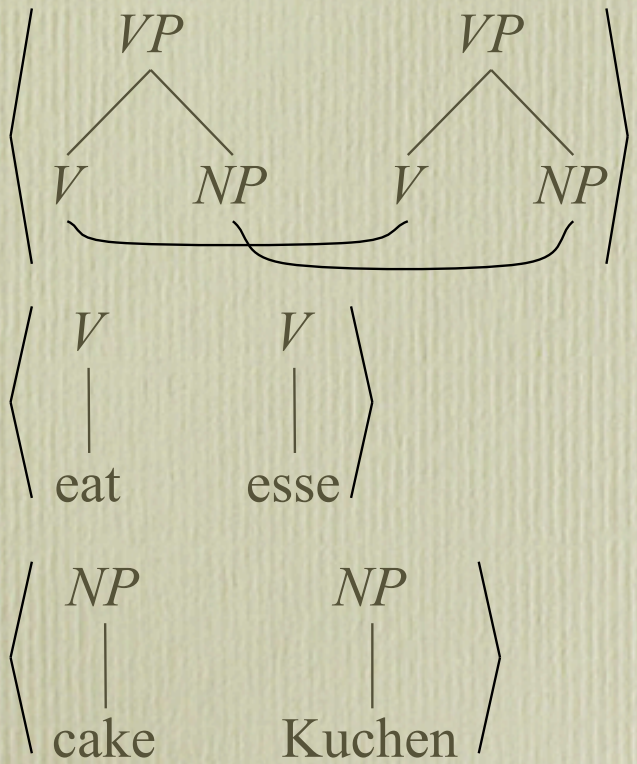
- = synchronous tree substitution grammars

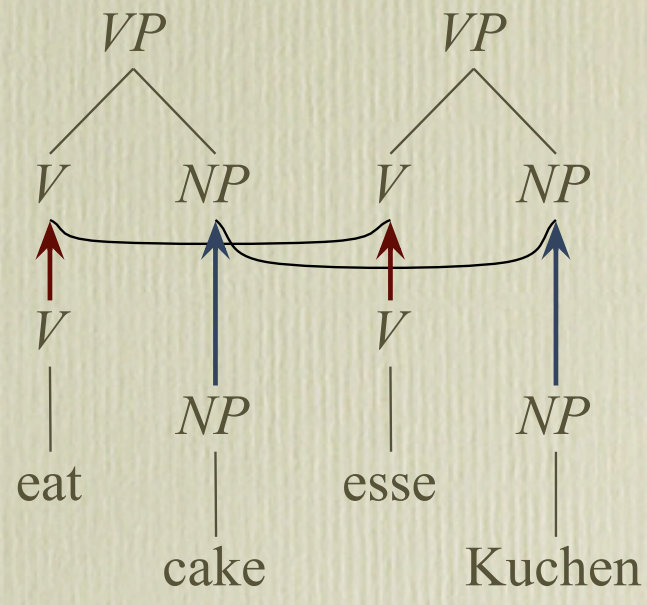
Synchronous Grammars

Context-free grammars as tree substitution

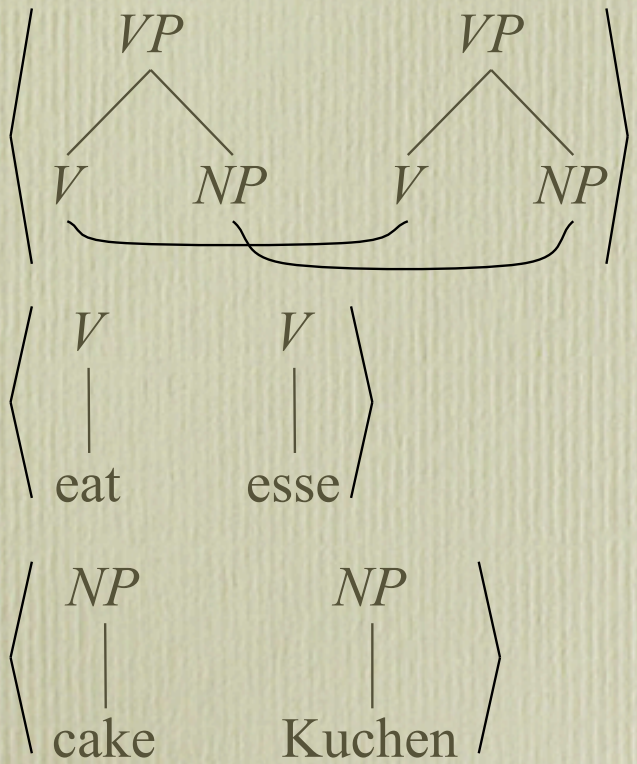


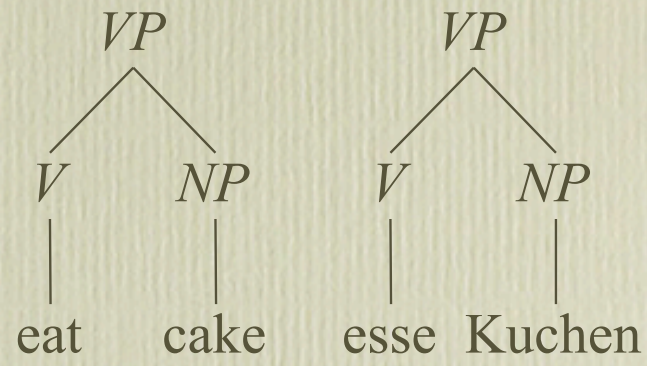
Synchronous Context-Free Grammars





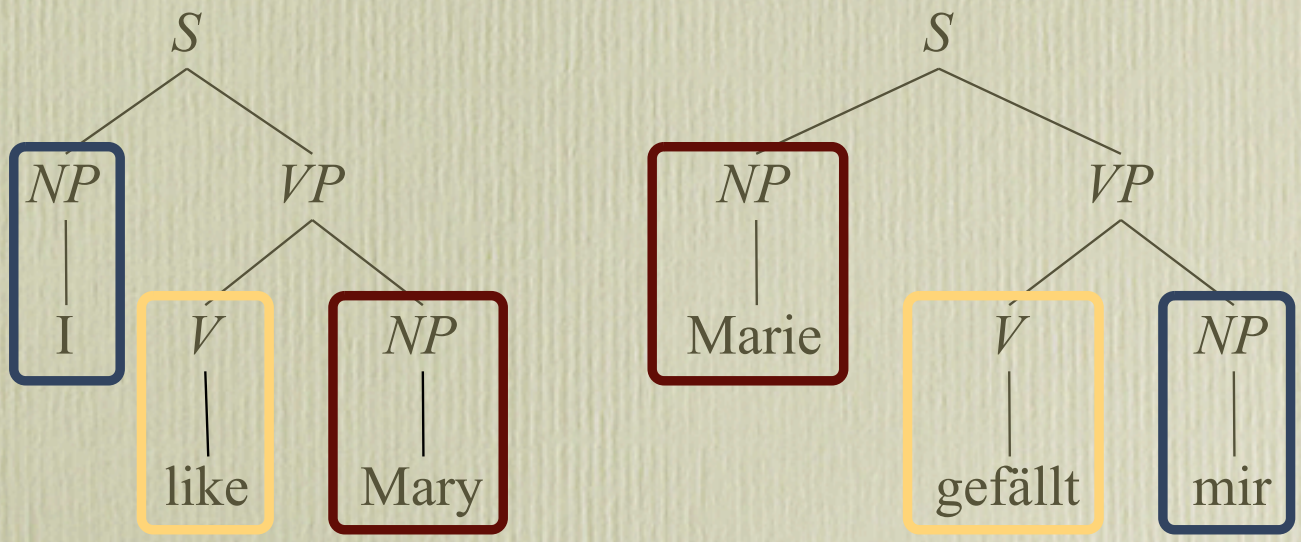
Synchronous Context-Free Grammars





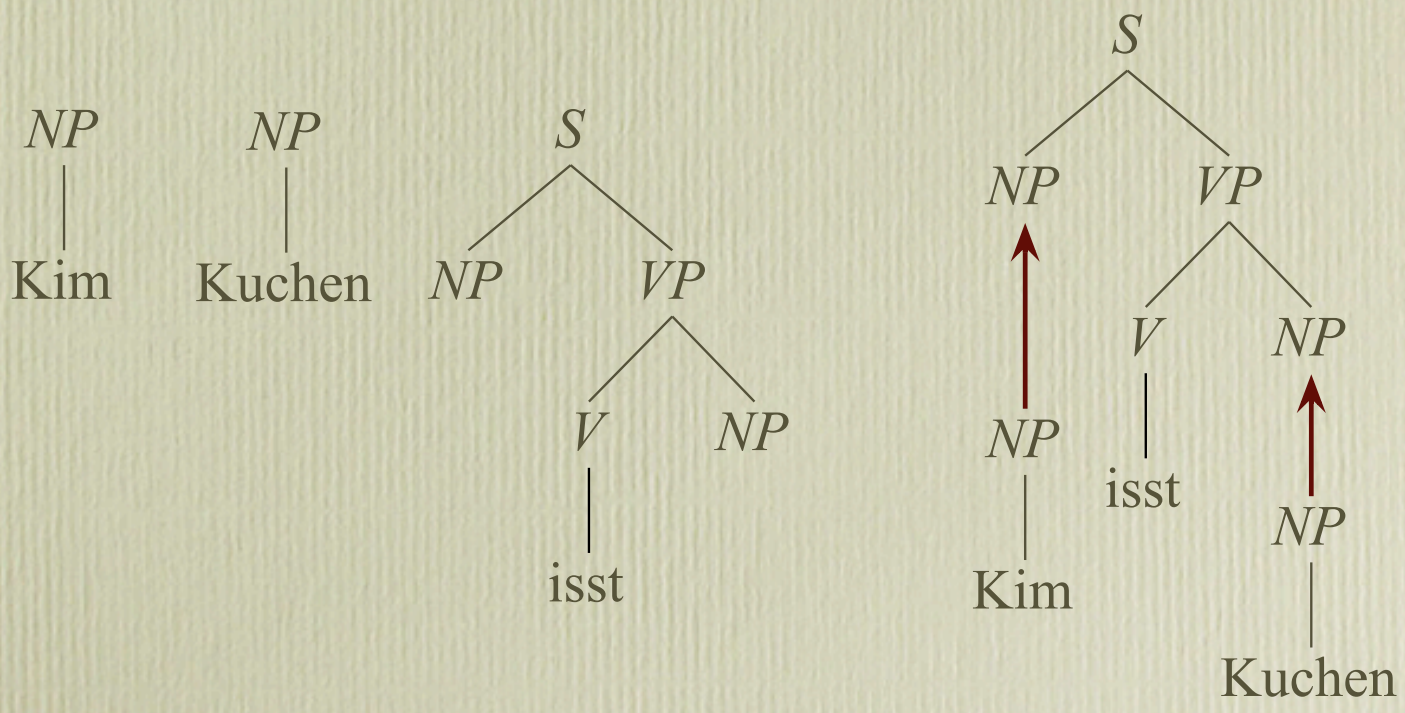
Problems With SCFG

Domain of locality is too small



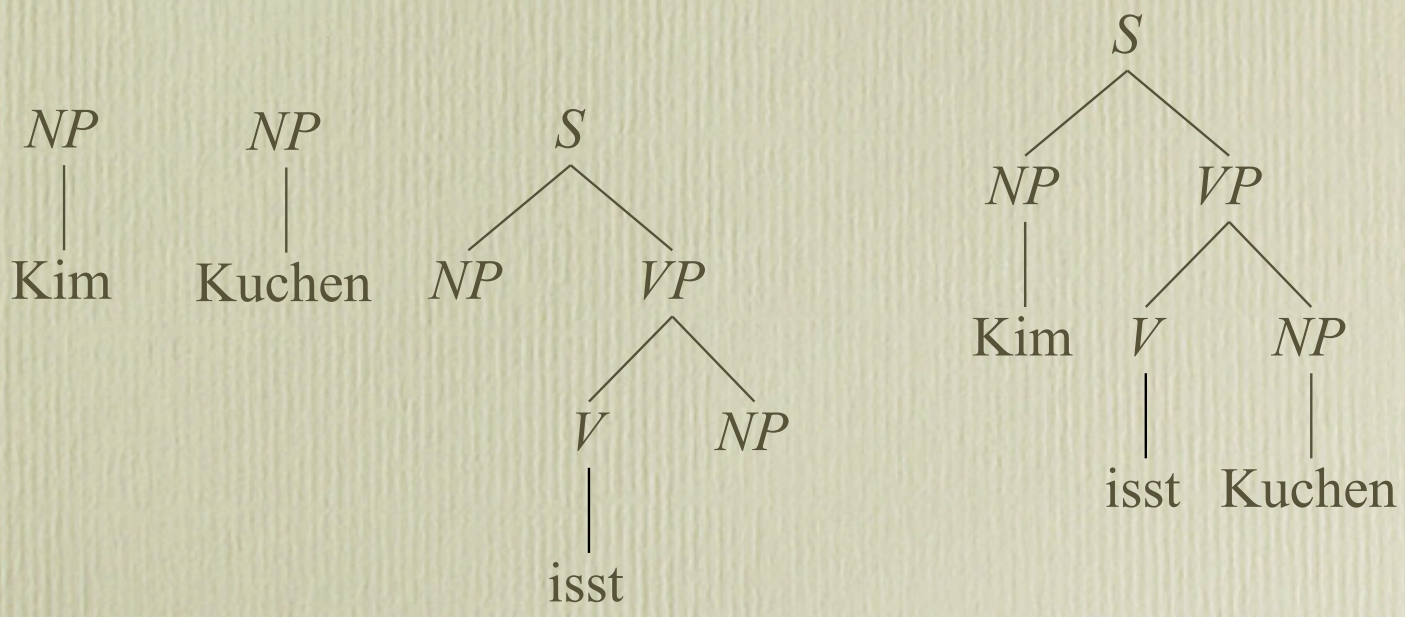
Tree Substitution Grammars

Expands domain of locality to *elementary tree*.



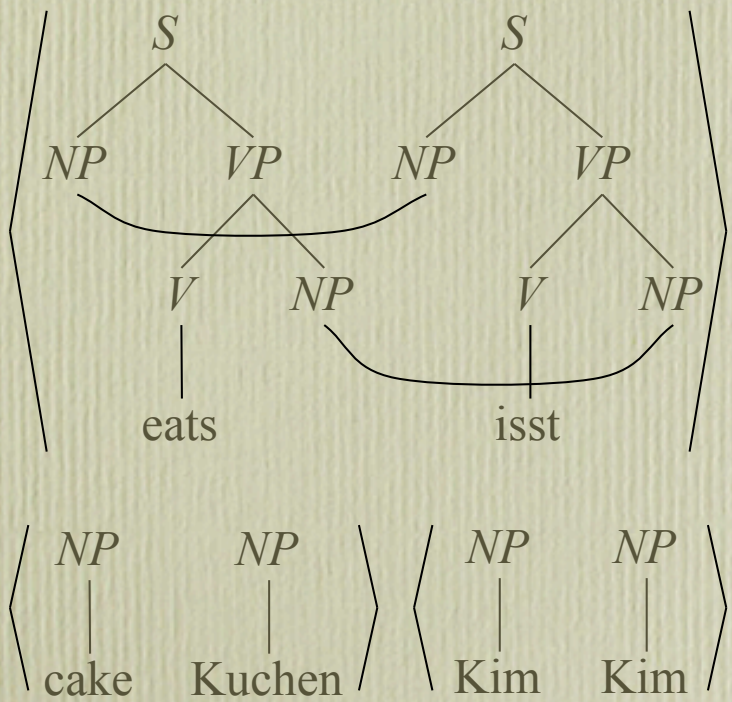
Tree Substitution Grammars

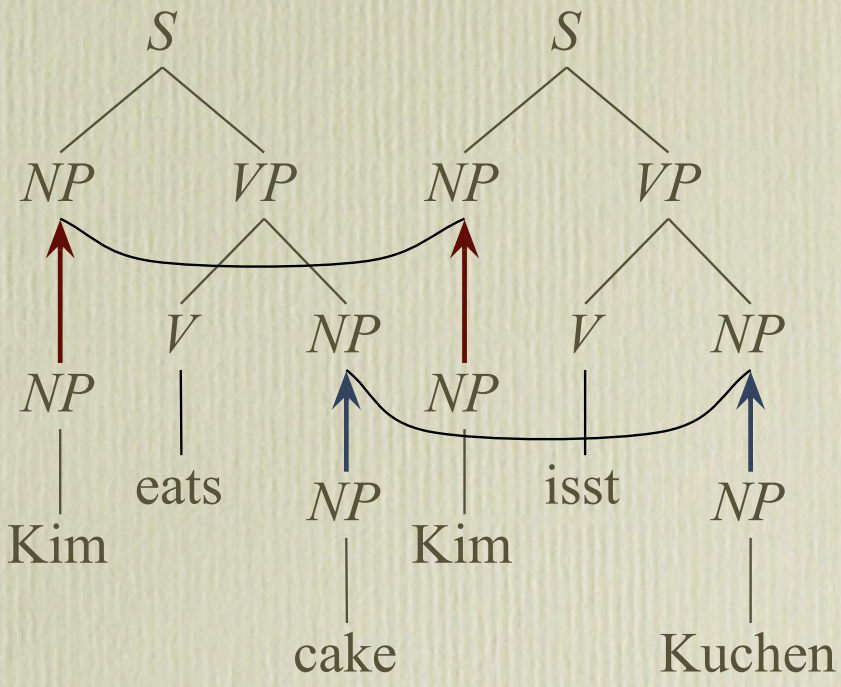
Expands domain of locality to *elementary tree*.



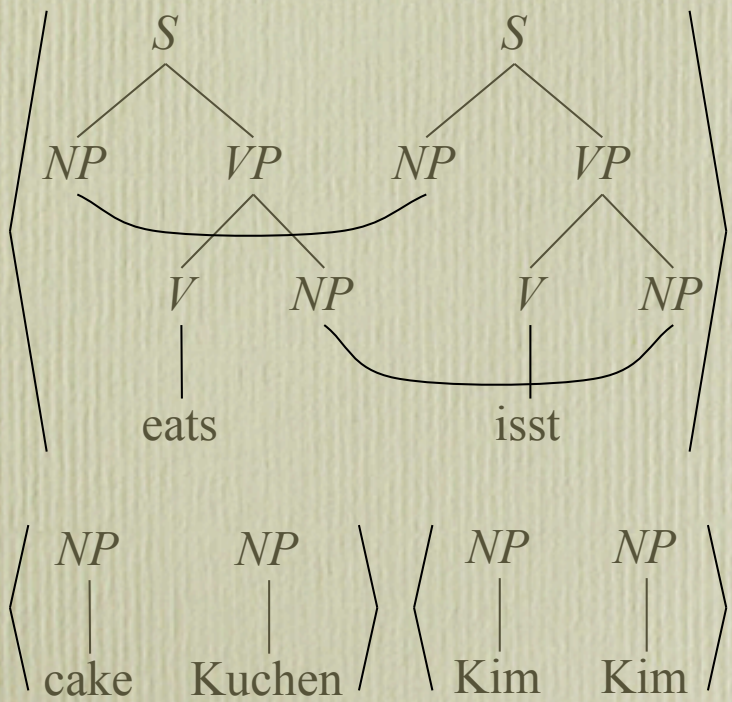
No additional expressive power over CFG.

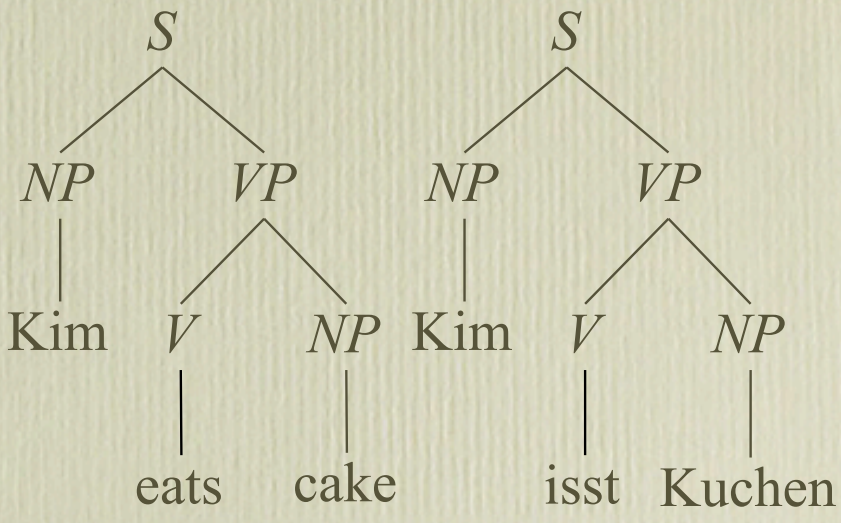
Synchronous Tree-Substitution Grammars



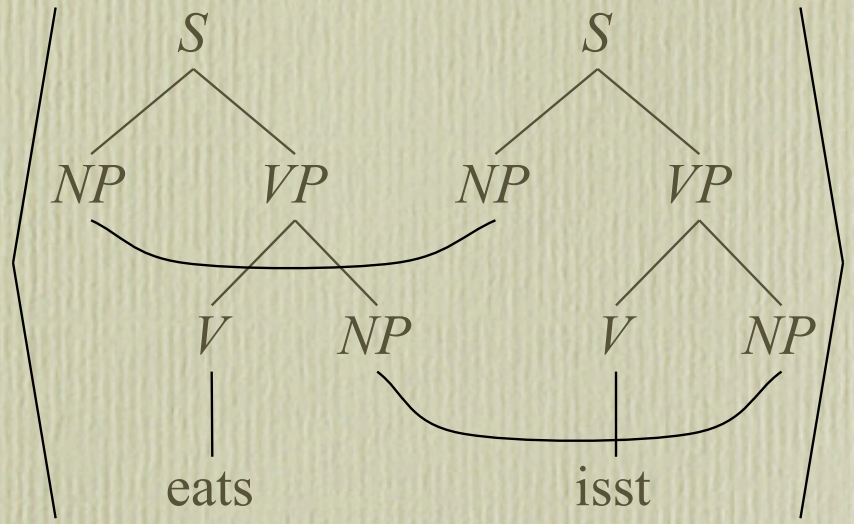


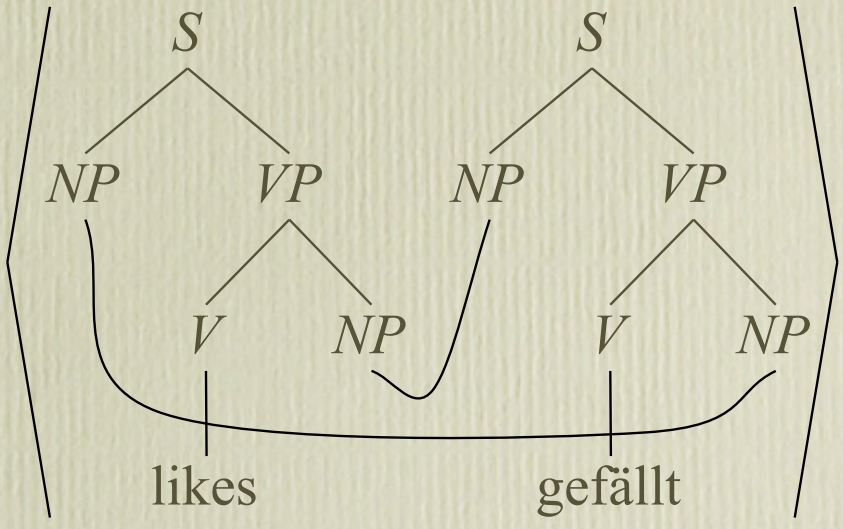
Synchronous Tree-Substitution Grammars



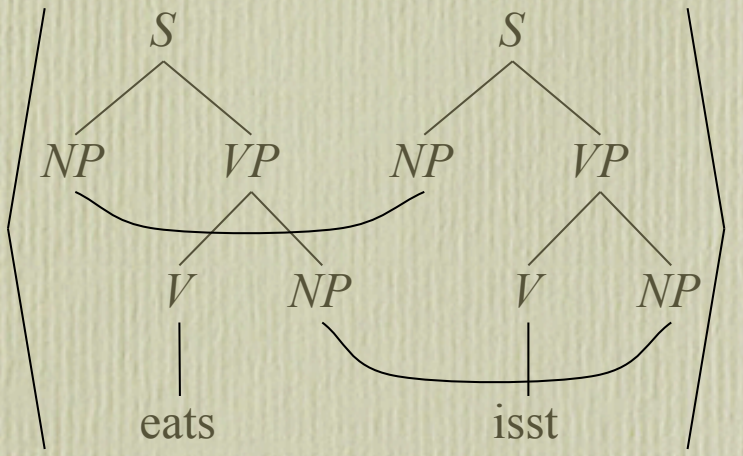


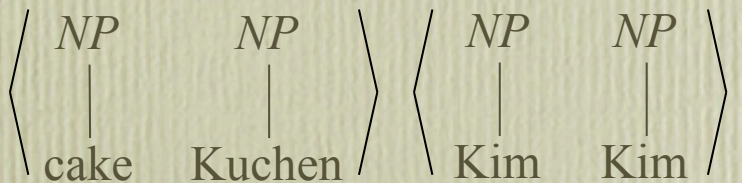
The STSG Payoff





STSG as Bimorphism





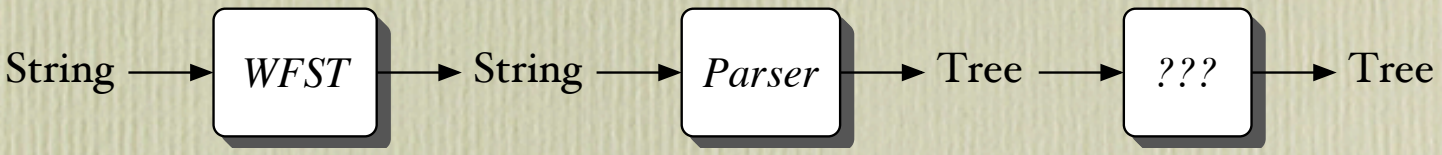
$$h_i(\alpha_1) = S(x, VP(V(eats), y)$$
$$h_o(\alpha_1) = S(x, VP(V(isst), y)$$
$$h_i(\alpha_2) = NP(cake)$$
$$h_o(\alpha_2) = NP(Kuchen)$$
$$h_i(\alpha_3) = NP(Kim)$$
$$h_o(\alpha_3) = NP(Kim)$$

$$h_i(\alpha_4) = S(x, VP(V(likes), y)$$
$$h_o(\alpha_4) = S(y, VP(V(gefällt), x)$$
$$h_i(\alpha_5) = NP(I)$$
$$h_o(\alpha_5) = NP(mir)$$

NB: linear, complete

Conclusions

Formalisms for the UNLP



For strings: weighted finite-state transducers

For trees:

- tree transducers: profligate growth, uninvertible
- linear tree transducers: no rotation
- bimorphisms unify transducer view and synchronous grammar view
- synchronous grammars may be in the right direction
  - Scansoft back-end