Computerlinguistik
WS 2018/2019

Greg Kobele

October 16, 2018

1 Overview[1]

In this course, we will study *automata theory* and *programming* as they relate to the following fundamental problems in linguistics

1. *well-formedness* of linguistic representations
2. *transformations* from one representation to another

1.1 Linguistic Theory

Linguistics is the theory of how we learn and use language. Like all theories of complex domains, linguistics approaches these complex phenomena by looking for the parts out of which they are composed. One basic idea is that we have an underlying *competence*, which is sometimes masked by non-linguistic factors (like cognitive limitations, etc). By competence, people usually mean knowledge of the regularities implicit in language.

1.1.1 Well-formedness

One fundamental job of linguistic theories (qua theories of our knowledge of various aspects of language) is to distinguish between *well-formed* (grammatical, acceptable, etc) and *ill-formed* (ungrammatical, unacceptable, etc) representations.[2]

[1]This handout adapted from one of Jeff Heinz.

[2]Although I used the words 'grammatical' and 'acceptable' interchangably above, they refer to very different concepts. An expression is 'grammatical' if it is generated/licensed by some grammar (usually thought to be the grammar in your head), whereas an expression is 'acceptable' if it is judged as such by a speaker. In other words, grammaticality is a statement about a theoretical property of an utterance, and acceptability is an empirical claim.

Strings In English, we can invent new words like **bling**. What about the following?

- **gding**
- **ngilb**
- **spilf**

Trees A compound word like "greenhouse" is a noun (a kind of house), rather than an adjective (a kind of green). What about the following:

- **dry-clean**
- **over-throw**
- **beer-drinking**

1.1.2 Manipulating Structure

Linguistic theory is also concerned with *transformations*. Transformations are used both to connect linguistic domains (mapping phonology to phonetics, or morphology to phonology, etc), as well as to describe the generalizations within a single domain.

Strings In phonology, underlying representations of words are transformed into surface representations.

- $dog + s \mapsto dogz$
- $matS + s \mapsto matSes$

Trees In syntax, trees are transformed into others by means of operations. In early theories, entire sentences were transformed into others.

- The dog ate my homework
  - My homework was eaten by the dog
  - What did the dog eat

1.2 Automata Theory

Automata are *abstract* machines, which we can use to answer questions like the following.

The Membership Problem

Given A (possibly infinite) set of strings (or trees) X

Input A string (or tree) x

Problem does x belong to X?

The membership problem is an abstract but formal recasting of the well-formedness question; take X to be the set of well-formed structures.[3]

The Transformation Problem

Given A (possibly infinite) mapping from strings/trees to strings/trees $f : X \to Y$

Input A string (or tree) x

Problem What is *f(x)*?

The transformation problem is an abstract (but formal) recasting of, well, the transformation question farther above.

1.2.1 Kinds of Automata

There are many kinds of automata. Two common types address these specific problems.

Recognizers Recognizers solve the membership problem

Transducers Transducers solve the transformation problem

Different kinds of automata can do different kinds of things. For example, restricting the amount of *memory* that an automaton can use quite expectedly limits what it can do.

Finite State Automata An automaton is finite-state if the amount of memory necessary to solve a problem for an input x is **bounded by a constant function** of the size of x

Linear Bounded Automata An automaton is linear-bounded if the amount of memory necessary to solve a problem for an input x is **bounded by a linear function** of the size of x

In this class we will focus on finite-state recognizers and transducers. There are many types of these as well.

- deterministic vs non-deterministic

[3]If, as is usual, there are infinitely many well-formed structures, we can take X to be given in some compact way (i.e. as a grammar).

- 1-way vs 2-way (for strings)
- bottom-up vs top-down vs walking (for trees)

The simplest type is the deterministic 1-way recognizer for strings. We will begin there, and then complicate things a little at a time.

1. add non-determinism
2. add outputs (transducers)
3. add 2-way-ness
4. generalize to trees and repeat

1.2.2 What do automata mean for linguistic theory?

While formal investigation of some domain can be interesting in its own right, one might wonder whether it is all 'general abstract nonsense', or whether there is some concrete benefit. One thing that automata give us is a well-defined notion of a *hierarchy* of machines; clearly, a machine with *fewer* restrictions on its memory can do *more* than one with *more* restrictions. We can thus identify a classification of problem instances into those which can be solved by automata with a certain kind of memory limitation. As it turns out, this automata inspired hierarchy seems to be highly relevant for linguistics.[4]

Fact 1 Finite-state automata over strings are sufficient for phonology

Fact 2 Finite-state automata over strings are **not** sufficient for syntax, but linear bounded automata are

Fact 3 Finite-state automata over trees are sufficient for syntax

Hypothesis Linguistic phenomena can be modeled by finite-state automata with even more restrictive memory requirements

[4]Perhaps this is to be expected, as the brain is a computing device, and automata provide a model of computation.