Massachusetts Institute of Technology
6.863J/9.611J Natural Language Processing, Spring, 2003
Department of Electrical Engineering and Computer Science
Department of Brain and Cognitive Sciences

Handout 1: Notes 1: Introduction to Computational Linguistics

Handed Out: February 5
I. Administrivia.

II. Agenda:

- What is computational linguistics? NL and the *computer representation* of *knowledge*. Levels of representation.
- What is parsing? A definition, and difficulty of parsing.
- Laboratory 1: morphology.

1 Levels of representation

In studying any complex system like language it is wise to keep in mind a variety of perspectives. Just as there is no single right description of an animal—we could describe organs, cells, digestive system, or enzymes—language can be understood on a number of different levels.

What do we mean by the information structure of language? Our rich and varied knowledge about language can be illustrated even by a simple sentence like *the cats ate the ice-cream*:

- We know how each word sounds and whether it's an English word at all. *Ca* begins a valid English word, but no English word will start out *ptk*. Further, the *s* on *cats* marks it as plural.
- We know that the words must appear in a certain order. *Cats the the ice-cream ate* doesn't say the same thing.
- We know that the sentence can be roughly divided into two parts familiar from grammar school: *the cats*, which is the *Subject* of the sentence, and *ate the ice-cream*, which is the *Predicate*. So the sentence's words are divided into distinct parts, or *constituents*. (These are also called the sentence's *phrases*.)
- We know who did what to whom: *the cats* is the agent of the action *ate* while *the ice-cream* is the thing eaten, or the affected object.
- We know that it's possible for cats to eat things, and we can picture what this means. The sentence would seem odd if it were *the paper clip ate the ice-cream*.
- We know that *the cats* refers to some group of particular cats, rather than cats in general.

- We know that the two sentences *John claimed the cats ate the ice-cream* and *John denied the cats ate the ice-cream* are logically incompatible.
- We know whether the sentence is *true* or not—perhaps whether in some particular situation the cats did indeed eat ice-cream. Put another way, we know the relationship between the sentence and the world—the sentence's *meaning*.
- We know that the sentence would sound fine if it were a part of the following two-sentence conversation, but odd if it introduced a conversation by itself: *I had fish for dinner last night. The cats ate the ice-cream.*

All this knowledge is clearly useful if we want to answer questions like who ate the ice-cream, draw inferences or analogies about what happened, or simply tell a joke about cats—in short, if we want to understand the sentence and use language competently.

Each knowledge source or description is in fact a *level of representation* or a *structural description*. That means three things. First, distinct information sources see the same sentence projected through different colored lenses—in terms of sounds, sound pieces, words, word pieces, who is the agent of the sentence, and so on. These information sources have a life of their own. The description of *cats* as a string of sounds doesn't say what cats can eat or whether the cats are the agents of the action. A good representation highlights exactly what needs to be said at a certain level of description while suppressing irrelevant detail. Put another way, a level of representation has its own atomic *primitives*—sounds like *k* in one case, words in another, or elements like *agent* and *action*.

Necessarily each kind of representation must be *partial*—because each tells us only *part* of what we know about the sentence. In the extreme of course, a particular representation might be so partial that it says nothing at all about a particular sentence—if the sentence were French instead of English, for example, then a non-French speaker might not be able to assign *any* representation of who did what to whom. In the view of parsing presented here then, there can be no such thing as an "ill-formed" sentence, simply sentences for which one can construct a more or less complete set of structural descriptions.

Second, the primitives can be put together in certain ways but not others. The word sequence *ice-cream like cats* is different from *like cats ice-cream*. For this reason, we will often call a representation a *structural description*, because it is a description that can be constructed in certain ways but not others.

Third and finally, distinct structural descriptions may be related to others. For instance, word order often matters in the description of a sentence in terms of the agent and the affected object. *Schank killed the spider* and *the spider killed Schank* clearly have very different implications about who gets affected, yet these two sentences contain the very same words, just in a different order.

Here we will briefly just list some of these different structural descriptions for sentences, mostly to introduce terminology.

1. *Sound structure* includes *phonetics* and *phonology*. Phonetics studies the actual pattern of speech sounds or *phones* roughly as they occur in speech. Phonology analyzes the sound pattern rules in a language, grouping together phones that behave the same under certain rules even if they sound different.

2. *Word structure*, or *morphology* and *morphophonemics* (after *morphos*, shape) analyzes how words are formed from minimal units of meaning, or *morphemes*; thus *cats* is the root *cat* and the plural morpheme *s*. Laboratory 1 explores this further.

3. *Phrase structure syntax*, sometimes called simply *syntax* (literally from the Greek *syntaxis*, $\sigma\upsilon\nu\tau\alpha\xi\iota\zeta$, meaning "arranged together") describes possible word combinations or *phrases* and how phrases are hierarchically arranged into sentences. For instance, in familiar terms the sentence *the cats ate the ice-cream* may be divided into two phrases: *the cats* forms a Subject phrase while *ate the ice-cream* is a Predicate Phrase. Laboratory 2 looks at the computational analysis of phrase structure in depth. We will also use the term *Surface structure* or S-structure, for an enriched phrase-structure representation.

4. *Thematic structure* or *case structure* casts sentences in a broad "who did what to whom" form, using notions like *agent*, *theme*, *affected object*, *recipient* of the action, and the like. Note how this ties to syntax: lexical properties affect what patterns we can have. For example: *the guy left*; *the guy solved the problem*; vs. *the guy solved*; *this problem, the guy solved the problem*. These are *verb subcategories*; the constraint is that each verb must discharge all its thematic roles, and all participants must receive a thematic role, somehow: the **theta criterion**. The level of **D-structure** we take to be a representation of the thematic relations. This is systematically related to an actual surface form, or *S-structure*, as we shall see.

5. *Lexical-conceptual structure* looks at words in terms of simple physical, causal elements and object classes, such as the representation of *ate* as meaning a thing moving along a path from one place to another (which happens when ice-cream moves into a cat's stomach). Laboratories 3, 4, and 5 look at phrase structure and thematic structure from statistical and modern linguistic points of view.

6. *Meaning structure* or *Semantic structure* covers both thematic structure and lexical-conceptual structure but also much more. It of course depends on what one takes "meaning" to mean. One popular view associates the meaning of a sentence with precisely those conditions that make the sentence *true*—its *truth conditions* (Tarski, Frege). On this view, language is a relation between meaning and the world, and so the meaning of *the cats ate the ice-cream* is identified with conditions C such that *the cats ate the ice-cream* is true if and only if conditions C hold. Clearly if we can represent C, we can draw inferences about *the cats*, paraphrase it, and so forth—all the things that one would normally associate with meaning.

   We use syntactic structure as a scaffolding on which to compositionally determine truth conditions. (Note that this is similar to what is done in programming languages using the model of *syntax-directed translation*.) Laboratory 6 explores this topic.

   Let's sketch how it works. To connect language to the world, we use the standard notion of an *interpretation* that systematically associates objects in the natural language to objects in a model world. We use set theory to build a *model* M in which particular sentences are true. So for example, we have words and syntactic phrases in our language, built according to specific rules. Corresponding to these, in the Model we have set-theoretic objects, and semantic rules for combining set-theoretic objects. (This is also

the same idea in *denotational semantics* for programming languages.) The set-oriented view is often very useful for natural language database query situations.

Here is a simple example. Let A be a set of individuals, entities in a model world M. Suppose there are just common nouns, like *cats*, *dogs*; determiners like *the*, *a*, and single thematic role verbs (intransitive predi-cats) like *sits* and *eats*. Now, word categories in our language can correspond to set-theoretic objects built out of A. E.g., *cat* is associated with the subset of A that contains cats; *eats* is associated with the subset that contains eaters. Nouns and Verbs are just subsets of A in the model; Determiners (*the*, *a*, etc.) are binary relations on subsets of A, that is, determiners are relations between properties (Frege).

Now we let $[\![a]\!]$be the denotation or "the interpretation of" a. Note that this objects are just sets. We define the interpretation of *the* to be $[\![\text{the}]\!]$= the binary relation $\text{THE}(x, y)$, where $\forall x, y$, $\text{THE}(x, y)$ is true iff $x \subseteq y \wedge |x| = 1$. The denotation of nouns is just the sets that correspond to them, e.g., $[\![cats]\!]$=$\{x$: x is a cat in $M\}$ Similarly for verbs. Now, to map between syntax and semantics, we need to have a (rule-for-rule) correspondence between the syntactic and semantic structures, with interpretations of syntactic objects on one side, and corresponding set relations on the other:

(1) (i) $[\![[_{\text{X}}\ \alpha]\ ]\!]= [\![\alpha]\!]$

(ii) $[\![[_{\text{VP}}\ \text{V}]]\!]= [\![\text{V}]\!]$(Denotation of single word or verb leaf is just that set of individuals.)

(iii) $[\![[_{\text{NP}}\ \text{Det N}]]\!]= \{\text{Y:}[\![\text{Det}]\!]([\![\text{N}]\!], \text{Y})\}$ (NPs are families of sets that stand in the Determiner relation to the set associated with the main noun.)

(iv) $[\![$[ntype:S NP VP]$]\!]$is true if $[\![\text{VP}]\!]\in [\![\text{NP}]\!]$, o.w. false. (S is true in model M iff set associated with VP falls into family of sets associated with subject NP.)

Now we can evaluate the truth conditions of a sentence like *the cat sleeps* by first evaluating the denotation rules at the leaves of the corresponding syntactic tree, and then combining the resulting sets as per the semantic combination rules until we get a truth condition for the whole sentence (with respect to M):

(2) (i) The interpretation of *the*, *cat*, and *sleeps* are simple:

(ii) $[\![the]\!]=$ THE;

(iii) $[\![[_{\text{N}}\ \text{cat}]]\!]= [\![cat]\!]=\{\text{x: x is a cat in } M\}$;

(iv) $[\![[_{\text{VP}}\ \text{sleeps}]]\!]= \{\text{x: x is a sleeper in } M\}$

(3) Now we combine the Det and N, to get a family of sets, Y: $[\![[_{\text{NP}}\ \text{the cat}]]\!]=$ $\{Y\text{:THE}([\![cat]\!], Y)\}= \{Y$: $\{\text{x:x is a cat}\}\subseteq Y \wedge |\{\text{x:x is a cat}\}|=1\}$ (the family of cats with at least 1 cat)

(4) The truth conditions for the whole sentence: if VP denotation, the set of sleepers, is a subset of subject NP denotation (i.e., their intersection is nonempty), i.e., the sentence is true iff exactly one cat that is also a sleeper, the intuitive result. (What about if *cats*?)

We will look at a more complex example later to see how the constraint on syntax, logical form, and semantics (model interpretation) can interact online.

Often meaning structure includes a level of *logical form* that explicitly represents the properties needed to properly evaluate such logical statements. Such forms generally include *variables*, hence the notion of *variable binding*—what set of things the variable can correspond to—as well as the *scope of a variable*, that part of the logical form to which a variable can be linked. (In our example, y may be linked to the second occurrence of *the* but not the first, thus y's scope mirrors the structure of the logical form.)

Pragmatic and *discourse structure* describe how language is used across sentences. People organize what they say into multiple utterances. At least on occasion people talk to each other, taking turns in conversation. Ritual interchanges like *please* and *thank you* are common. This activity too is part of language, and we can ask what the atoms of this descriptive level are, how they are put together, and how they relate to other structural descriptions. Pragmatic structure may include a different concept of meaning that does not affect the truth conditions of a sentence—what is dubbed its *implicatures*. For example the sentence, *The cats ate the ice-cream, but Elissa still likes them.* might not hold in the world (it might not be true that if the cats ate the ice-cream then I still like them, no matter what), but Elissa still might *mean* what she says. The structural descriptions required here often include a complete reasoning system about knowledge, belief, causes, and intentions, what we know and what we think others think we know—something that extends into our general abilities to reason. The line between language and thinking blurs quickly indeed.

It is conventional to divide the study of language into two parts: syntax, the study of word arrangements without regard for meaning; and semantics, the study of meaning. This division has been the source of much controversy. From the structural description point of view it is a needless border dispute. Sentences and even linguistic descriptions don't come labeled "syntactic" or "semantic." All structural descriptions—sound structure, lexical-conceptual, phrasal, and logical—are syntactic, in the sense that they allow some combinations of primitives and not others. Further, there is no unitary field of semantics. There are only different structural descriptions, each of which are ultimately related to meaning in different ways. When we use the term *syntax*, it will always be clear from context just what kind of syntax is meant—what structural description is being constructed.

To summarize, a particular structural description or level of representation for a sentence has three components:

1. A proprietary, primitive *vocabulary*;
2. A *syntax*, or constraints on how to glue atoms and strings of atoms into bigger pieces;
3. A (possibly empty) *systematic relation* to other structural descriptions for that sentence.

It is the business of linguistics to ask what these linguistic representations should look like.

2 Multiple levels of representation and the parsing problem

If structural descriptions tell us *what* we know about a sentence, then *how* is that knowledge used? Even though we have barely touched upon what knowledge of language is, one can already define the central *how* question of computational linguistics, which will occupy us in the rest of this course. This the *parsing problem*:

> **The parsing problem:** Given a sentence (or set of sentences) compute some or all of the structural descriptions associated with that sentence (or set of sentences).

Figure 1 illustrates the parsing problem for a sentence like *the cats ate the ice-cream*. The sentence lies along one axis, like the spine of a spiral notebook, and the different descriptions hang off it like the notebook's leaves. Separately and collectively these structural descriptions are called a *parse* of the sentence. Our goal is to develop explicit algorithms to mimic the effect of the projection lines in the figure.

This approach raises a whole host of questions. Putting aside the crucial questions of what the descriptions are and how they are put together, we have not said a word about how the different structural descriptions are related; in figure 1 the different descriptions coexist in parallel. Presumably, a sentence might not even have a particular structural description. This would be true, for instance, if the sentence was malformed in some conventional way, such as *the cats—the cats they ate ice-cream*. Then one could possibly still recover enough of the structural descriptions to figure out what the sentence means. Then too, it could be that the structural descriptions that say what we know about language are only indirectly related to the structural descriptions that people actually use or that we design computers to use. Nor have we said *how* these representations are computed. Are they to be built up one at a time? In some particular order? How are they to be combined? Do they act like joint constraints, so that we can solve the solution(s) to structural descriptions like a set of simultaneous equations? Because we've used the word *compute* we have implied that the procedure used to recover structural descriptions is *effective*, or computable. Is this even possible? Can it be done efficiently?

3 Computation and Parsing

It is important to note that in this book parsing means recovering *any* kind of structural description, not just the familiar grammar-school notions like subject and predicate. It is even a broader picture than what's usually found even in computer science, where parsing means simply chopping up a sentence into constituents like *the cats* and *ate the ice-cream*.

Plainly parsing depends on what linguistic representations are adopted. Different theories can emphasize completely different structural descriptions or even exclude descriptions that other theories take as essential. For example, some approaches omit the influence of word order and word groups on meaning, instead relying on the cues provided by individual words to recover descriptions of who did what to whom. An important goal of computational linguistics is to find the right descriptions that aid language processing.

Parsing is possible because the levels of representation are systematically connected. Thematic constraints tell us about syntactic phrases: if we run into *sleep*, we shouldn't expect a

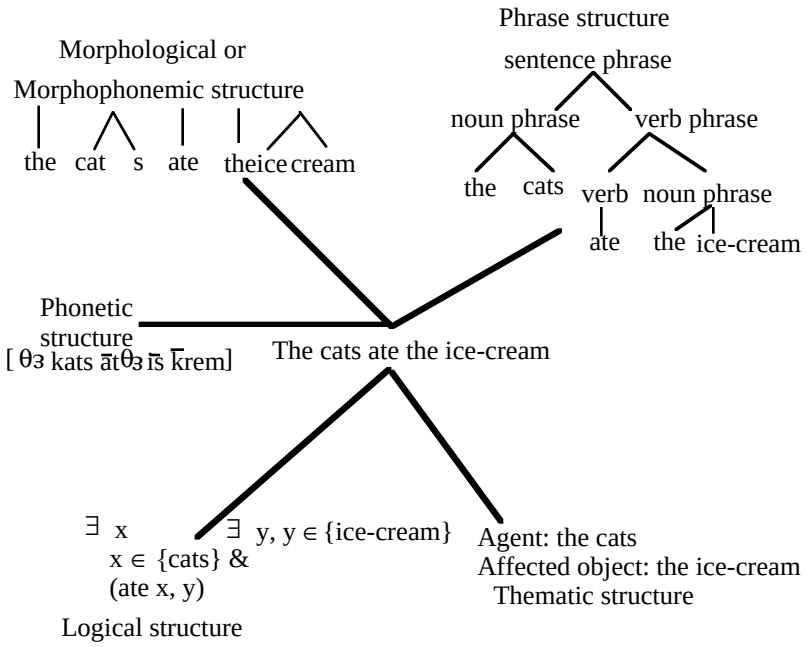


Figure 1: The "spiral notebook" picture of language description and parsing. The sentence *the cats ate the ice-cream* forms the backbone of the notebook. Different structural descriptions are projected from the spine, including the sound structure of the sentence, its word structure, a hierarchical tree structure of its phrases like Subject and Predicate, and so forth. Collectively these representations form a *parse* of the sentence. If two or more structural descriptions of the same type are projected from the same sentence, we say that the sentence is *ambiguous*. See the next section for more discussion about the topology of this picture.

thematic role next. Similarly, syntactic configurations help fix interpretation, as we have seen. **Important:** the levels of representation view is agnostic about the *temporal* order of computation. It should be obvious that, given a sentence, we *need not wait* until all the syntactic phrases are built before we start interpreting them. We can use any piece of information at any time—semantic information or verb subcategories can constraint syntactic analysis, and vice-versa. In fact, it is crucial to do this to gain computational efficiency.

Of course, it should come as no surprise that in situations where the usual cues drop out there must be other constraints that take up the slack, else communication would falter. So we find that newspaper editors cramped for headline space drop some of the usual telltale structure as well, producing sequences like *Executive raises double workers'* whose meaning must be pieced together some other way. After all, the whole point of a headline is to jam a few keywords into limited space, so it should not be surprising that in impoverished information contexts like these we draw on alternative constraints and representations. Some leaves of the spiral notebook will be obscured or missing. By design, impoverished examples can illuminate one page or another, but not the whole notebook.

What makes parsing a challenge is *ambiguity*.

Definition 1: A sentence S is *ambiguous* if there exists more than one structural description of a given type associated with S.

In terms of our spiral-notebook figure, this just means that the mapping from a typical sentence back to any one structural description is in general one-to-many.

Ambiguity is pervasive in natural language. It can occur at *every* level of structural description. For example, the sentence *every cat ate the ice-cream on a table* has many kinds of ambiguities.

At the level of sounds, *ate* could project either as *ate* or *eight*. At the level of words *cream* can be either a noun or a verb (on one calculation, about 40% of words are ambiguous in this kind of way in running text). At the level of syntactic phrases, *on the table* could be either the location of the ice-cream or the location of the cats. At the level of meaning structure, there is *word sense ambiguity* (*cats* could be either animals or people). At the level of thematic structure, the same verb can have more than one set of thematic roles: e.g., *walk* can be intransitive or not, as in *John walked to the store*, *John walked Bill to the store*. Further, *a table* could have its scope *outside* the word *every*, meaning that there was one table on which every cat ate, or, things could be the other way around, so that there would be more tables for more cats— for every cat there was some (perhaps different) table on which that cat ate.

Since the projection backwards from the sentence we are given to its structural descriptions can in general be one to many, this makes it hard to unravel what the right description is. Parsing therefore involves *search*. A key goal of computational linguistics is to reduce or eliminate search, by reducing ambiguity, avoiding the explicit representation of all possible ambiguities (perhaps by delaying their computation), or, a related strategy, avoiding repeated computation by factoring multiple ambiguous representations into one.

For example, a typical "factoring" approach to *the cats ate ice-cream on the table* might represent the ambiguity this way, as triples of phrases:

(5)

(S 0 6)	(NP 0 2)
(VP 2 6)	(NP 3 6)
(NP 3 4)	(PP 4 6)
(NP 5 6)	

Note how the PP and the NP *the table* get built just once, even though they participate in two parses. This is the key to efficient natural language analysis.

4 Parsing and Morphology

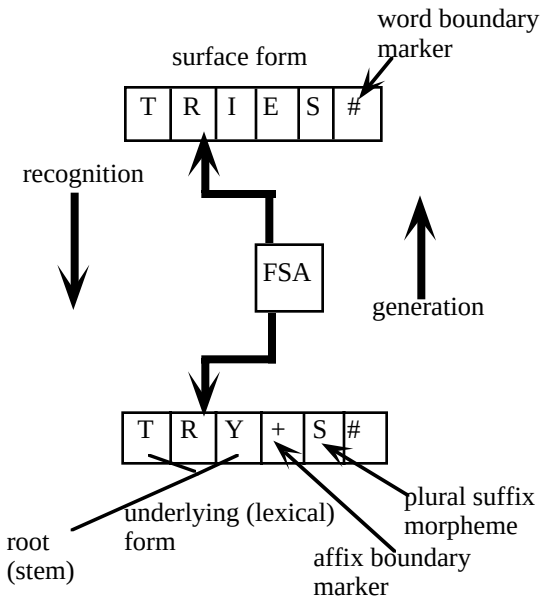
Morphological parsing demonstrates *all* the problems of full-blown parsing. Plus, it's a key part of all NL systems.

What is the problem? Given a surface form, e.g., *tries*, we want to look it up as *try+s* (Why?)

The key questions to ask about morphological analysis are the usual ones: first, what is the *knowledge* we must encode; second, how should we *represent* that knowledge; third, how can we *compute* with that representation.

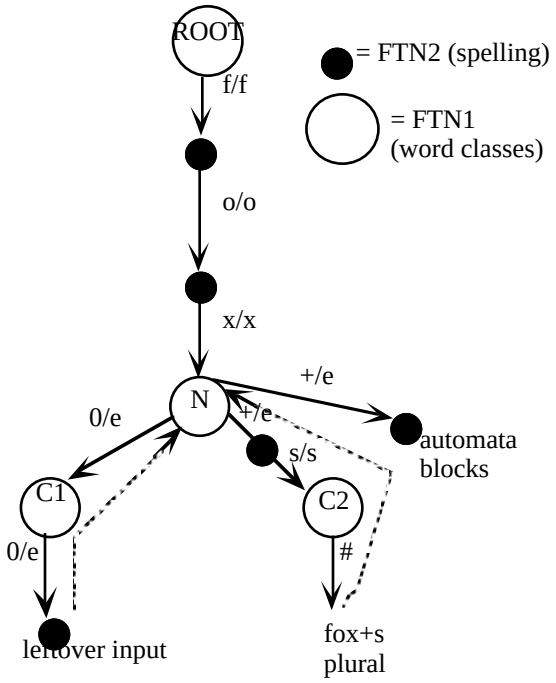
What knowledge? Plainly any system needs to know: (1) what roots (stems) and affixes (prefixes/suffixes) there are; (2) what affixes precede/follow what stems; (3) how spelling can change appearances when we glue stems and affixes together. (Thus this assumes the simplest kind of linear, concatenative syntax; is this enough for all languages? What about *foot/feet*?

What representation? The most obvious and simplest is "what you see is what you get": a linear string for the surface form like *tries*, and a linear string for the underlying or *lexical* form, plus a mapping between them. That's the KIMMO system.



- FSA1: Encodes spelling changes as a finite-state transducer

- FSA2: Encodes root+suffix combinations as series of root, suffix (lexicon) classes. Example classes for English: noun, verb, noun-singular, noun-plural, progressive, agentive, comparative. Each state of this automaton encodes a different root+suffix class, like plural nouns vs. singular nouns, which could allow different endings.



Notation: The KIMMO system uses the following terminology.

morpheme= a minimal unit of meaning

affix= morpheme added to *root* or *stem* of a word, e.g., *s* to form plural. *Suffix*= added at end; *prefix*= at beginning; *infix*=inside word (e.g., *sit–sat*)

surface form = how a word is normally spelled

underlying form, *lexical form* = spelling form used to look up word in dictionary

lexicon= when referring to the KIMMO system, a state in FSA2 corresponding to a root or suffix class.

alternation class= a label made up of one or more *lexicon classes*, used to summarize what lexicon (FSA2) states might follow an actual dictionary word.

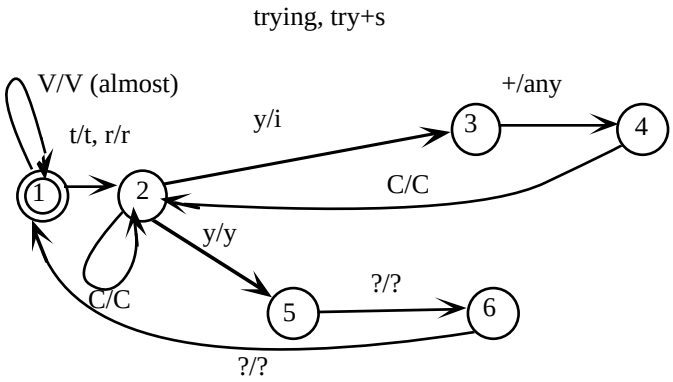
a *continuation class*= a label for FSA2 not covered as an alternation class

+ = root boundary marker

ϵ= null (empty) character

= end of input marker

5 A spelling change example



To add:
i/i, a/a, y/y (in state 1)

6 Handling multiple spelling changes

Apply all changes in parallel (or used merged fsa). (FTN intersection) Merger can sometimes require the cross-product of automaton states, but usually not. (Why is it the cross-product?)

Spelling rules for English. 5 rules plus 1 identity rule (required for implementation). A "full" rule set might include 50+ rules, but some systems have used up to 350 rules.

Rule	Example	# states
1. Epenthesis (EP)	fox–foxes; cat–cats	6
2. Gemination (G)	cool–cooler; big–bigger	16
3. Y-spelling (Y)	toy–toys; try–tries	6
4. Elision (EL)	large–larger	15
5. I-spelling (I)	lie–lying	7

In English a regular verb takes the following endings:

Category	Ending	Example	Abbreviation
First person, *I*	∅	*I like*	
Second person, *You*	∅	*You like*	
Third person, *it*	+*s*	*It likes*	P3
First person, plural *We*	∅	*We like*	
Past	+*ed*	*liked*	PS
Past Participle	+*ed*	*were liked*	PR
Progressive	+*ing*	*liking*	PP
Agentive	+*er*	*liker*	AG
Able	+*able*	*likable*	AB

The abbreviations P3, PS, PR, PP, AG, and AB are dictionary divisions that we need because, for example, a past tense verb *liked* takes a different ending from a progressive *liking*. Besides these divisions, we need others because certain verbs take some of these endings but not others. First of all, there are two main classes of irregular verbs: those that can just take the regular progressive, agentive, and able endings, but is otherwise irregular (an example is *do*, since we have *doing* and *doable* but not *doed*); and those verbs that take the third person singular, progressive, and agentive and able endings (an example is *bite*, since we have *bites* and *biting* but not *bited*).

Call the first *i*rregular *v*erb class IV1, and the second IV2. Further, there are verbs that appear just in third person, like *is* (class IP3); verbs that are irregular past forms, like *kept* (class IPS); and irregular past participles, like *bitten* (Class IPP). Adding these up, we have 6 verb dictionary divisions: regular verbs, V; IV1; IV2; IP3, IPS, and IPP. In addition, the class V leads to 7 possible choices, because a verb could have one of 7 possible endings: P3, PS, PP, PR, I(irregular), AG, and AB. Each of these acts like the state in a finite-state device, to prepare the lookup procedure for what might come next.

7 An example

In the trace that follows, we'll keep track of the transduction letter-by-letter. Indentation will indicate progress through the string, so when we backup the trace will reflect this by unindenting that line. The number in the left-hand column will mirror this indentation. The spelling change automata are traced in parallel, via a 6-element vector. Each number in the vector is the current state of that automaton. Below each part of the 6-element vector we'll indicate the abbreviation of the relevant automaton, EP, G, Y, EL, or I. The first element of the vector is always the identity (default) automaton. Each step will show the pair of characters being processed in (underlying, surface) form. The symbol 0 is used to denote the empty string. We'll now step through the recognition of *races'*, one letter at a time.

```
Recognizing surface form "races'".
0 (r.r) --> (1 1  1 2 1  1)
               EP G Y EL I

1   (a.a) --> (1 1  4 1 2  1)
                 EP G Y EL I

2     (c.c) --> (1 2 16 2 11 1)

3       (e.0) --> (1 1 16 1 12 1)
                     EP G Y EL I

4         Entry |race| ends --> new lexicon N, config (1 1 16 1 12 1)
                                                        EP G Y EL I
```