Grammars

- A formal definition of the syntactic structure of a language
- Set of rules that tell you whether a sentence is correctly structured
- A sentence is a well-formed string in the language
- Production rules specify the order of components and their sub-components in a sentence
- Top-level rule
  - one production rule designated as the “start rule”
  - provides the structure for an entire sentence

Production rules

- Production rules specify a syntactic category and assign to it a sequence of zero or more symbols
- Symbols are either terminal or non-terminal
- Terminal symbols
  - correspond to components of the sentence with no internal syntactic structure
- Non-terminal symbols
  - any symbol assigned values by a production rule

Sentence parsing and generation

- A grammar can be used to parse a sentence or to generate one
- Parsing begins with a sentence and ends with the top level rule
  - at lowest level sentence is composed of terminal symbols, first assign a terminal syntactic category to each component
  - assign non-terminal symbols to each appropriate group of terminals, up to the level of entire sentence
- Generation starts from the top-level rule and chooses one alternative production wherever there is a choice

Example

- The English grammar
  - a set of rules for combining words into well-formed phrases and clauses
### Clauses

- A clause is a group of related words containing a subject and a verb
  - building blocks of sentences
- Independent clause
  - can stand alone as complete sentence
  - *Tammy is great!*
- Dependent clause
  - unable to stand alone as complete sentence
  - “Although Tammy is great” is a dependent word: allows clause to be embedded in another sentence
  - *Although Tammy is great, this class is still boring.*

### Phrases

- A phrase is a group of related words that does not contain a subject-verb relationship
  - can consist of a single word or a group of words
- Sentences formed by noun-phrases and verb-phrases
- Noun-phrases can be of the form
  - noun
  - article noun
  - article adjective noun
- Examples:
  - *Dogs*
  - *Some dogs*
  - *The mean dogs*

- Verb-phrases can be of the form
  - verb
  - verb adverb
- Examples
  - *bite*
  - *drool profusely*
- A noun-phrase can be contained in a verb-phrase
  - *bite people*
  - the noun *people* is the object of the verb *bite*

### A simple English grammar

- Consider independent clauses only
- Non-terminal symbols are indicated by angle brackets
- Start rule
  - `<SENTENCE> => <NOUN-PHRASE> <VERB-PHRASE>`
- Production rules
  - `<NOUN-PHRASE> => <NOUN> | <ARTICLE> <NOUN> | <ARTICLE> <ADJECTIVE> <NOUN>`
  - `<VERB-PHRASE> => <VERB> | <VERB> <NOUN-PHRASE>`
- These rules specify all non-terminal symbols
  - we haven’t specified the terminals yet

### Terminal symbols

- Still need to specify terminal symbols for our grammar
- In particular, for the non-terminal symbols
  - `<NOUN>`
  - `<VERB>`
  - `<ARTICLE>`
  - `<ADJECTIVE>`
- Let’s assign them as follows:
  - `<NOUN> => DOG | CAT | WATER`
  - `<VERB> => BIT | SMelled | DRANK | SCRATCHED`
  - `<ARTICLE> => THE | A`
  - `<ADJECTIVE> => STINKY | HAPPY | MEAN`
Sentence generation

- According to our grammar, the following sentences are syntactically correct:

  - DOG BIT
  - DOG BIT CAT
  - THE CAT BIT THE DOG
  - THE HAPPY DOG SNIFFED THE STINKY CAT
  - A MEAN CAT SCRATCHED THE DOG
  - A STINKY DOG DRANK THE WATER
  - THE HAPPY WATER DRANK THE STINKY DOG
  - THE MEAN WATER SCRATCHED THE HAPPY CAT

Generation

  - THE HAPPY DOG SNIFFED THE STINKY CAT
  - THE MEAN WATER SCRATCHED THE HAPPY CAT

• How many different sentences can be generated by this grammar?
• A: 3024 (27 possible noun-phrases multiplied by 112 possible verb-phrases)

Generation

  - THE HAPPY WATER DRANK THE STINKY DOG

• Process of taking a sentence and fitting it to a grammar

  - Grammars are used to recognize syntactically correct sentences
  - Example

    THE HAPPY DOG SNIFFED THE STINKY CAT

    Fit the above sentence to the given grammar:

    - NOUN => DOG | CAT | WATER
    - VERB => BIT | SNIFFED | DRANK | SCRATCHED
    - ARTICLE => THE | A
    - ADJECTIVE => STINKY | HAPPY | MEAN

Parsing

- Process of taking a sentence and fitting it to a grammar

  - Parsing English is complex due to context dependence
  - Natural language understanding is one of the hardest problems of artificial intelligence
    - human language is complex, irregular and diverse
    - philosophical problems of meaning
### Parsing example

THE HAPPY DOG SNIFFED THE STINKY CAT

\[ \langle \text{SENTENCE} \rangle \Rightarrow \langle \text{ARTICLE} \rangle \ \langle \text{ADJECTIVE} \rangle \ \langle \text{NOUN} \rangle \ \langle \text{VERB} \rangle \]

\[ \langle \text{NOUN-PHRASE} \rangle \ \langle \text{VERB} \rangle \ \langle \text{NOUN-PHRASE} \rangle \]

\[ \langle \text{SENTENCE} \rangle \]

### Optional parts of speech

- The given grammar provided options for phrases

\[ \langle \text{SENTENCE} \rangle \Rightarrow \langle \text{ARTICLE} \rangle \ \langle \text{NOUN} \rangle \]

\[ \langle \text{ARTICLE} \rangle \ \langle \text{ADJECTIVE} \rangle \ \langle \text{NOUN} \rangle \]

\[ \langle \text{NOUN-PHRASE} \rangle \ \langle \text{VERB} \rangle \ \langle \text{NOUN-PHRASE} \rangle \]

\[ \langle \text{VERB-PHRASE} \rangle \]

\[ \langle \text{SENTENCE} \rangle \]

### Syntax and semantics

- Syntax only tells you if the sentence is constructed correctly
- Semantics tells you whether a correctly structured sentence makes any sense
- The sentences

THE HAPPY WATER DRANK THE STINKY DOG

THE MEAN WATER SCRATCHED THE HAPPY CAT

are correct syntactically but something appears wrong ...
- WATER usually isn’t happy or mean and usually doesn’t drink or scratch either

### Formal specifications

- Need a precise notation of syntax of a language
  - grammars can be used for generation and parsing
- Context-free grammars

\[ \langle \text{name} \rangle \Rightarrow \text{sequence of letters and/or digits that begins with a letter} \]

\[ \langle \text{name} \rangle \Rightarrow \text{gikB} \]

\[ \langle \text{name} \rangle \Rightarrow \text{msg42} \]

- Substitute as many times as necessary
- All legal statements can be generated this way

### Context-free grammar

- person = firstname + " " + lastname;

- How do we get this from our grammar?

- Unlike natural languages such as English, all the legal strings in a programming language can be specified using a context-free grammar

### Recursive Sentence Generator (RSG)

- Constructs sentences, paragraphs, and even papers that fit a prescribed format
- RSG demo applet, courtesy of Prof. Forbes, Duke CS
  - http://www.duke.edu/web/cps001/code/RSG.html
- The format is specified by a user-defined grammar
- You will define your own grammars in Lab 9
- Some example grammars are here:
  - http://www.duke.edu/web/cps001/code/grammars/
- We will go over the poem grammar (Poem.g)
RSG syntax

- Production rules enclosed in curly braces ({})
- Nonterminals are enclosed in angle brackets
- Terminals are plain text
- First line of production rule specifies syntactic category
  - options are separated by semicolons
- Must specify top-level rule with nonterminal <start>

```plaintext

{ 
  <start>
  your top level rule here. can be as many sentences as you like. ;
}

Poem.g

{ {The <object> <verb> tonight. ;} {<object> waves ; big yellow flowers ; slugs ;} {<verb> sigh <adverb> ; portend like <object> ;} {<adverb> warily ; grumpily and <adverb> ;} }

Poem.g

- Top-level rule specifies one sentence with 2 nonterminals

```plaintext

{ 
  <start>
  The <object> <verb> tonight. ;
}

- Nonterminal <object> provides three options, all terminal

```plaintext

{ 
  <object>
  waves ;
  big yellow flowers ;
  slugs ;
}

Poem.g

- Nonterminals can refer to other nonterminals and be combinations of terminals and nonterminals
- Nonterminal <verb> refers to the nonterminals <adverb> and <object>

```plaintext

{ 
  <verb>
  sigh <adverb> ;
  portend like <object> ;
}

- Nonterminal <object> is already defined
- Need to define <adverb>

Poem.g

- Nonterminals can refer to themselves
- Nonterminal <adverb> has two options
  - first is terminal
  - second refers to <adverb>

```plaintext

{ 
  <adverb>
  warily ;
  grumpily and <adverb> ;
}

- What would happen if there was no terminal option?

Generating a poem

- All sentences start with <start>
- Only one production in the definition of <start>
The <object> <verb> tonight.
- Expand each grammar element from left to right
  - <start> is a terminal, so it is simply printed
  - <object> is a non-terminal, so it must be expanded
    Choose one:
      - waves
      - big yellow flowers
      - slugs
    Suppose slugs is chosen
**Generating a poem**

The slugs `<verb>` tonight.
- `<verb>` is a non-terminal, so it must be expanded
  Choose one:
  - `sigh `<adverb>`
  - `portend` like `<object>`
  Suppose `sigh `<adverb>` is chosen
The slugs sigh `<adverb>` tonight.
- `<adverb>` is a non-terminal, so it must be expanded
  Choose one:
  - `warily`
  - `grumpily`
  Suppose `warily` is chosen

**A complete poem**

The slugs sigh warily tonight.
- The terminal `tonight` is simply printed
- There are no more non-terminals to expand!
- The grammar has generated a complete poem

**Question:**
- Why is this called a recursive sentence generator?

---

**More poems**

- Go to the RSG demo applet and select the poem grammar to generate more poems
  [http://www.duke.edu/web/cps001/code/RSG.html](http://www.duke.edu/web/cps001/code/RSG.html)
- How many different poems are possible?