Theory and Practice of Data Cleaning

Introduction to Regular Expressions: From **Theory** to Practice

Introduction to Regular Expressions (Regex)
Theory & Practice

- **Theory of regular expressions:**
  - Brief introduction where regular expressions come from …
- **Practice of regular expressions:**
  - What you need to know to get started with regex in practice!
- **Demonstration of regular expressions**

Why study regular expressions?

- Widely used **in practice**:
  - A bit like *wildcards* (e.g. find all csv file: `*.csv`)
  - … but much **more powerful** (“*wildcards on steroids*”) !

- Used to *match*, *extract*, *find-and-replace* data, e.g.,
  - ... in text editors
  - … scripting and programming (Bash, Python, Perl, R, ..., Java, ...)
  - ... screen scraping and other data extraction applications

Why study regular expressions for data cleaning?

- Useful to **match** (*assess*) and **transform** (*clean*) data:
  - **OpenRefine** Expression Language (*GREL*)
  - Use in scripting languages for **data cleaning**
  - … *workflow automation*

- **Example: *ISO 8601*** *date format*:
  - **YYYY-MM-DD**
- … vs *other (common) date formats*:
  - MM/DD/YY
  - DD.MM.YYYY
  - …

Introduction to Regular Expressions (Regex)
Theory & Practice

- **Theory of regular expressions:**
  - Brief introduction where regular expressions come from …
- **Practice of regular expressions:**
  - What you need to know to get started with regex in practice!
- **Demonstration of regular expressions**

Theory of Regular Expressions

- **Regular expression** (*regex*): in ***theoretical computer science*** (esp. *formal language theory*):
  - A formal expression that defines a ***search pattern***
  - ... used to ***match*** (or ***recognize***) a strings

Theory of Regular Expressions

- **Formal definition:**
- Base elements:
  - ∅ ***empty set***, ε ***empty string***, and Σ ***alphabet*** *of* characters
- Given regular expressions *R* and *S*, the following are also regular expressions:
  - *R* | *S* ***alternation***
  - *RS* ***concatenation***
  - *R** ***Kleene star***
  - *(R)* ***parentheses*** (can be omitted with *precedence rules*)

Regular Languages in the Chomsky Hierarchy

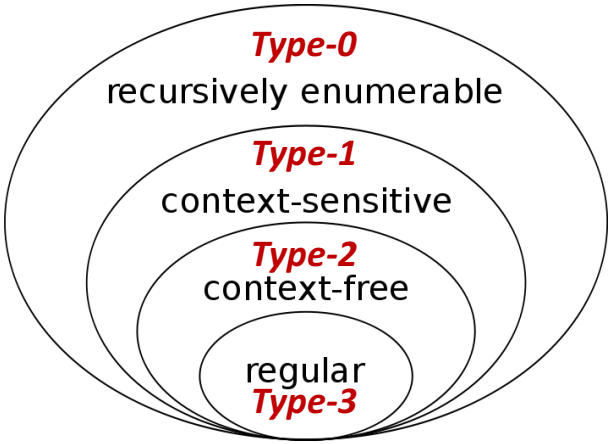
v • t • e **Automata theory: formal languages and formal grammars** [hide]

Chomsky hierarchy	Grammars	Languages	Abstract machines
Type-0	Unrestricted	Recursively enumerable	Turing machine
—	(no common name)	Decidable	Decider
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
—	Positive range concatenation	Positive range concatenation*	PTIME Turing Machine
—	Indexed	Indexed*	Nested stack
—	—	—	Thread automaton
—	Linear context-free rewriting systems	Linear context-free rewriting language	restricted Tree stack automaton
—	Tree-adjoining	Tree-adjoining	Embedded pushdown
Type-2	Context-free	Context-free	Nondeterministic pushdown
—	Deterministic context-free	Deterministic context-free	Deterministic pushdown
—	Visibly pushdown	Visibly pushdown	Visibly pushdown
Type-3	Regular	Regular	Finite
—	—	Star-free	Counter-free (with aperiodic finite monoid)
—	Non-recursive	Finite	Acyclic finite

Each category of languages, except those marked by a *, is a proper subset of the category directly above it.
Any language in each category is generated by a grammar and by an automaton in the category in the same line.

Categories: Formal languages | Finite automata

The "regular" in regular expression



https://en.wikipedia.org/wiki/Regular_language

Regular Grammars

Example: **floating point numbers** such as `-0.314159265e+1`
… can be **generated** by a ***right regular grammar G*** with
N = {**S**, A,B,C,D,E,F}, **Σ** = {0,1,2,3,4,5,6,7,8,9,+,-,.,e},

Production rules P =

S → +A	A → 0A	B → 0C	C → 0C	D → +E	E → 0F	F → 0F
S → -A	A → 1A	B → 1C	C → 1C	D → -E	E → 1F	F → 1F
S → A	A → 2A	B → 2C	C → 2C	D → E	E → 2F	F → 2F
	A → 3A	B → 3C	C → 3C		E → 3F	F → 3F
	A → 4A	B → 4C	C → 4C		E → 4F	F → 4F
	A → 5A	B → 5C	C → 5C		E → 5F	F → 5F
	A → 6A	B → 6C	C → 6C		E → 6F	F → 6F
	A → 7A	B → 7C	C → 7C		E → 7F	F → 7F
	A → 8A	B → 8C	C → 8C		E → 8F	F → 8F
	A → 9A	B → 9C	C → 9C		E → 9F	F → 9F
	A → .B		C → eD			F → ε
	A → B		C → ε			

Regular Grammars

Example: **floating point numbers** such as `-0.314159265e+1`
… can be **generated** by a ***right regular grammar G*** with
N = {**S**, A,B,C,D,E,F}, **Σ** = {0,1,2,3,4,5,6,7,8,9,+,-,.,e},

Production rules P =

S → +A	A → 0A	B → 0C	C → 0C	D → +E	E → 0F	F → 0F
S → -A	A → 1A	B → 1C	C → 1C	D → -E	E → 1F	F → 1F
S → A	A → 2A	B → 2C	C → 2C	D → E	E → 2F	F → 2F
	A → 3A	B → 3C	C → 3C		E → 3F	F → 3F
	A → 4A	B → 4C	C → 4C		E → 4F	F → 4F
	A → 5A	B → 5C	C → 5C		E → 5F	F → 5F
	A → 6A	B → 6C	C → 6C		E → 6F	F → 6F
	A → 7A	B → 7C	C → 7C		E → 7F	F → 7F
	A → 8A	B → 8C	C → 8C		E → 8F	F → 8F
	A → 9A	B → 9C	C → 9C		E → 9F	F → 9F
	A → .B		C → eD			F → ε
	A → B		C → ε			

- Not very handy in practice …
- **Regular expressions** to the rescue!

[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?