

Introduction to Computational Linguistics

Frank Richter

fr@sfs.uni-tuebingen.de.

**Seminar für Sprachwissenschaft
Eberhard Karls Universität Tübingen
Germany**

Regular Relations

- Regular expressions can contain two kinds of symbols: unary symbols and symbol pairs.
 - Unary symbols (a, b, etc) denote strings.
 - Symbol pairs (a:b, a:0, 0:b, etc.) denote pairs of strings.
- The simplest kind of regular expression contains a single symbol. E.g., “a” denotes the set {a}.
- Similarly, the regular expression “a:b” denotes the singleton relation {⟨a, b⟩}.
- A regular relation can be viewed as a mapping between two regular languages. The a:b relation is simply the crossproduct of the languages denoted by the expressions a and b.

Finite-State Transducer

Definition 10 (FST) A finite-state transducer is a 6-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ where

Σ_1 is a finite alphabet,
(called the *input alphabet*)

Σ_2 is a finite alphabet,
(called the *output alphabet*)

Q is a finite set of *states*,

$i \in Q$ is the *initial state*,

$F \subseteq Q$ the set of *final states*, and

$E \subseteq Q \times (\Sigma_1^* \times \Sigma_2^*) \times Q$
is the set of edges.

Constructing Regular Relations

- Crossproduct: $A \cdot x \cdot B$
 - The crossproduct operator, $\cdot x \cdot$, is used only with expressions that denote a regular language; it constructs a relation between them.
 - $[A \cdot x \cdot B]$ designates the relation that maps every string of A to every string of B . If A contains x and B contains y , the pair $\langle x, y \rangle$ is included in the crossproduct.

Constructing Regular Relations

- Composition: $A \circ B$
 - Composition is an operation on relations that yields a new relation. $[A \circ B]$ maps strings that are in the upper language of A to strings that are in the lower language of B .
 - If A contains the pair $\langle x, y \rangle$ and B contains the pair $\langle y, z \rangle$, the pair $\langle x, z \rangle$ is in the composite relation.

Properties of Regular Relations

Regular relations in general are *not closed* under

- complementation,
- intersection, and
- subtraction.

Properties of Transducers

- A transducer is functional iff for any input there is at most one output.
- A transducer is sequential iff no state has more than one arc with the same symbol on the input side.

Replacement Operators

- Unconditional obligatory replacement:

$$A \rightarrow B =_{def} [[\sim \$[A - []] [A .x. B]]^* \sim \$[A - []]]$$

- Unconditional optional replacement:

$$A (\rightarrow) B =_{def} [[\sim \$[A - []] [A .x. A | A .x. B]]^* \sim \$[A - []]]$$

- Contextual obligatory replacement:

$$A \rightarrow B \parallel L _ R$$

meaning: “Replace A by B in the context L _ R.”

Non-determinism of *replace* (1)

Example: $ab \rightarrow ba \mid x$

meaning: “replace ab by ba or x
non-deterministically”

Sample input: abcbba

Outputs: bacbbaa, bacbxa,
xcdbbaa, xcdbxa

Non-determinism of *replace* (2)

Example: $[a\ b \mid b \mid b\ a \mid a\ b\ a] \rightarrow x$

meaning: “replace ab or b or ba or aba by x ”

Sample input: aba aba a b a a b a

Outputs: x a axa a x x

Longest match, left-to-right replace

- For many applications, it is useful to define another version of replacement that in all such cases yields a unique outcome.
- The longest-match, left-to-right replace operator, $@->$, defined in Karttunen (1996), imposes a unique factorization on every input.
- The replacement sites are selected from left to right, not allowing any overlaps.
- If there are alternate candidate strings starting at the same location, only the longest one is replaced.

A Grammar for Date Expressions

1To9 = [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]

0To9 = [%0 | 1To9]

SP = [", "]

Day = [Monday | ... | Saturday | Sunday]

Month = [January | ... | November | December]

Date = [1To9 | [1 | 2] 0To9 | 3 [%0 | 1]]

Year = 1To9 (0To9 (0To9 (0To9)))

DateExp = Day | (Day SP) Month " " Date (SP Year)

Marking Date Expressions

- A parser for date expressions can be compiled from the following simple regular expression:
DateExp @-> %[... %]
- The above expression can be compiled into a finite-state transducer.
- @-> is a replacement operator which scans the input from left to right and follows a longest-match.
- Due to the longest match constraint, the transducer brackets only the maximal date expressions.
- The dots mean: identity with the upper string. The whole expression means: replace DateExp by DateExp surrounded by brackets.

Overgeneration Problem

- The grammar for date expressions accepts illegal dates.
- Example: It admits dates like “February 30, 2007”.
- More generally:
 - If a grammar admits strings that should not be accepted by the grammar, the grammar is said to *overgenerate*.
 - If a grammar does not admit strings that should be accepted by the grammar, the grammar is said to *undergenerate*.

Tokenizing Date Expressions

Example:

Today is [Wednesday, August 28, 1996] because yesterday was [Tuesday] and it was [August 27] so tomorrow must be [Thursday, August 29] and not [August 30, 1996] as it says on the program.

Incremental Tokenization

input layer one, two, and so on.

single word layer one || , || two || , || and || so || on || . ||

multi-word layer one || , || two || , || and so on || . ||

Advantages of Incremental Tokenization

- With finite-state transducers incremental tokenization is implemented by the composition operator for transducers.
- Separation of grammar specification and program code: Each analysis level is specified in a well-defined language of regular expressions.
- Transducers for each layer can be stated independently of each other.
- Regular expressions can be compiled automatically into (composed) finite state transducers.