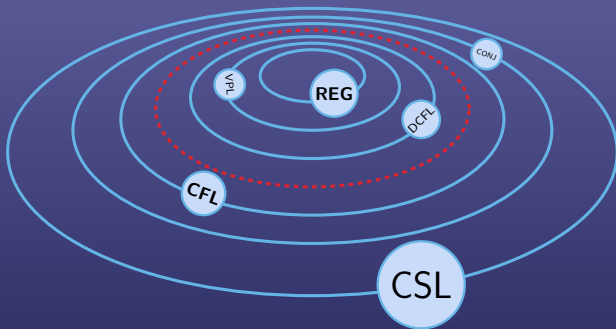




**Bauman Moscow State University**  
**Th. Computer Science Dept.**

# Introduction to Realms of Formal Languages



*Antonina Nepeivoda*  
*a\_nevod@mail.ru*

# Lecture Outline

## 1 Formal Languages by Examples

- Automata Around Us
- Tons of Formalizations

## 2 Course Details

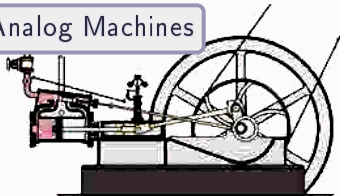
## 3 Basics on Term Rewriting

- Rewriting and Reduction
- Ordering and Induction
- String Rewriting



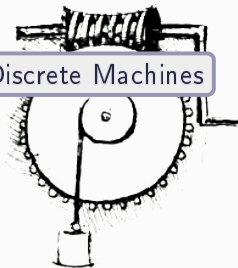
# Physical Concept of Automaton

## Analog Machines



Continuous control  
Differential equations

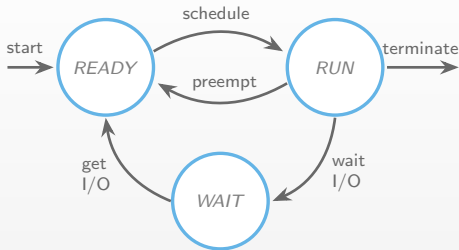
## Discrete Machines



Discrete control  
Algebra & Logics



# Automata Specifications



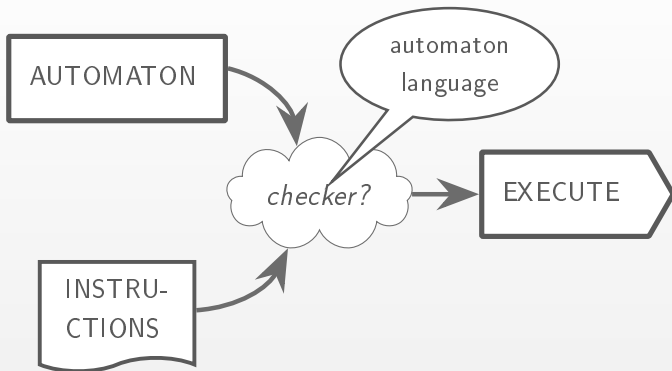
- CPU Scheduler scheme
- Automata-based schemes are used widely:
  - communication protocols;
  - user interfaces;
  - control units;
  - optimisation cycles.



- Simple game menu



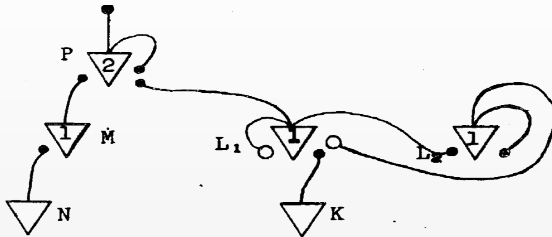
# Programming: Automata Languages



- One-by-one input steps  $\Rightarrow$  automaton model is enough.
- Sequence of multiple steps given simultaneously  $\Rightarrow$  executor is required to verify the sequence wrt the automata model.
- Admissible step sequences — *a formal language* of the machine.



# McCulloch & Pitts Neural Networks



● — excitatory signal;

○ — inhibitory signal;

▽ — an input neuron;

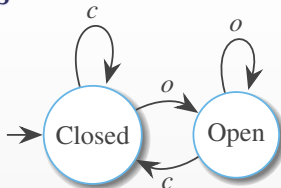
▽ $k$  — an inner neuron firing whenever none of the inhibitory signals and at least  $k$  of excitatory signals fire.

Kleene (1951) introduced regular languages, describing the events in McCulloch–Pitts NN in terms of three operations:  $\{., \cup, *\}$ .

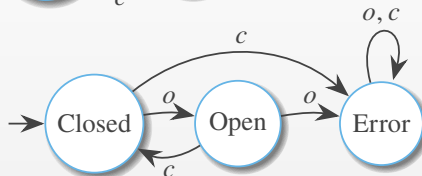


# Automata Models

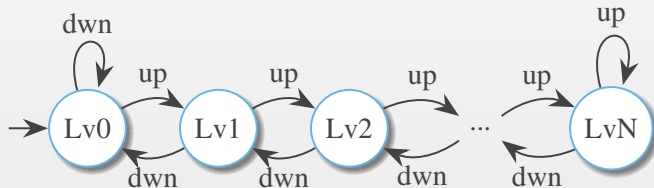
Door automaton:



Channel automaton:



Elevator:



# Formal Languages Formally

## 👁👁 Definition

Let us consider an algebra  $\mathcal{A} = \langle \underbrace{\mathcal{M}}_{\text{carrier}}, \overbrace{\mathcal{F}}^{\text{signature}} \rangle$ .

A formal language is a set  $\mathcal{M}$  of terms in algebra  $\mathcal{A}$ .

- Classical case: if  $\mathcal{A} = \langle \Sigma, \cdot \rangle$ , where  $\cdot$  is the concatenation operation, then  $\mathcal{M} \subseteq \Sigma^*$ , where  $*$  is iteration (Kleene star, Kleene closure).
- Wider scope: tree automata languages, syntax trees, process graphs...





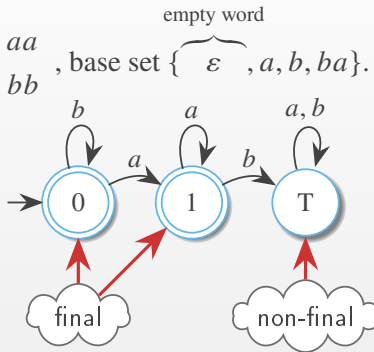
# Formal Languages Examples

- syntax {
- $\{\underbrace{aa\dots a}_{n \text{ times}} \underbrace{bb\dots b}_{n \cdot 3 \text{ times}} \mid n \in \mathbb{N}\}$  (henceforth  $\{a^n b^{3n} \mid n \in \mathbb{N}\}$ );
  - words containing a square number of letters  $a$   
 $\left\{w \mid \exists k ( \underbrace{|w|_a}_{\text{number of letters } a \text{ in word } w} = k^2 ) \right\};$
  - Russian palindromes of the even length  
(«Я сплнся, я сплнся, я сплнся»);
  - sequences of balanced parentheses;
  - well-formed arithmetic expressions over  $\mathbb{N}$  and  $\{\cdot, +\}$ .
- semantics {
- all tautologies in the classical logic;
  - all consistently typed programs in Python;
  - formal languages with linear-time-decidable membership.



# Varieties of FL Representations

- Set comprehension:  $\{w \mid w \text{ does not contain subword } ab\}$ .
- Algebraic expressions:  $b^* a^*$ .
- Term rewriting systems:  $\begin{cases} a \rightarrow aa \\ b \rightarrow bb \end{cases}$ , base set  $\{\overbrace{\varepsilon}^{\text{empty word}}, a, b, ba\}$ .



- Recognising machines:

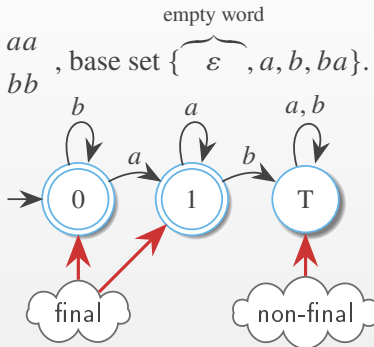
- First-order or second-order logical formulas:

$$\forall x, y (\underbrace{Q_a(x)}_{x \text{ is equal to } a} \ \& \ \underbrace{S(x, y)}_{y \text{ succeeds } x} \Rightarrow \underbrace{\neg Q_b(y)}_{y \text{ is not equal to } b}).$$



# Varieties of FL Representations

- Set comprehension:  $\{w \mid w \text{ does not contain subword } ab\}$ .
- Algebraic expressions:  $b^*a^*$ .
- Term rewriting systems:  $\begin{cases} a \rightarrow aa \\ b \rightarrow bb \end{cases}$ , base set  $\{\overbrace{\varepsilon}^{\text{empty word}}, a, b, ba\}$ .
- Recognising machines:

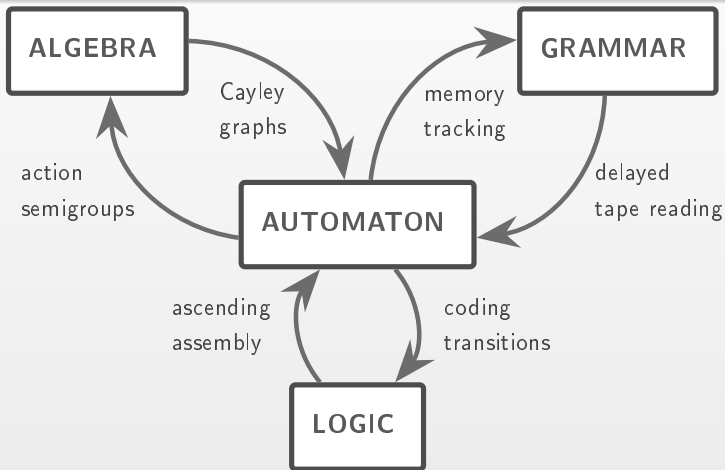


*Automata-based representation easily reduces to Turing machines  
 $\Rightarrow$  useful when estimating computational complexity of the  
language recognition.*



# Transforming & Analysing Representations

- *Efficient conversions between representation classes*
- *Constructing an optimal form of a representation inside a class*



# Spherical Cows in RoFL Course

What is a reason to study artificial languages like «words not containing subwords *ba*»?

---



(common knowledge):  
*A cow is homeomorphic  
to a sphere with a couple  
of handles.*

- Transducer images are shorter and their intermediate interpretations are more sheer;
- The transducer images reveal the essence of abstract formal properties rather than features of the ad-hoc model;
- In fact, morphisms and FSM transformations can be considered as lexers into the theoretical scope.



# Course Structure

- Introduction and TRS

- I part: automata
  - Finite Automata;
  - Visibly Pushdown Automata;
  - Generic Pushdown Automata;
  - Deterministic Pushdown Automata;
  - Alternating Automata, Memory Automata;
  - Cellular Automata, Tree Automata + *Midterm I*.
- II part: languages
  - Semirings;
  - Syntactic Monoids;
  - Ehrenfeucht–Fraïssé games and Pumping;
  - Hardest Languages and Language Representations;
  - Language Inference;
  - Combinatorial & Computational Language Properties.
- *Midterm II*
- Concluding Lecture
- «*RoFL Farm*»



## 👁👁 Score Arrangement

- *Midterms*  $\times 2 \times 15+$ .
- *Assignments*  $\times 5 \times 8+$ .
  - *Java, Python, Go, JS* — no score bonus
  - *C/C++, Kotlin, Dart, TypeScript, Lua* — +1 point
  - *Rust, Lisp dialects, Scala, Julia* — +2 points
  - *Haskell, Erlang* — +3 points
  - *Peφaλ* — +4 points for first time +3 afterwards
- **Deadlines for assignments:**
  - 0-14 days — no penalty
  - 15-21 days — 1 score penalty
  - 22-28 days — 2 score penalty
  - 29-∞ — 3 score penalty



# Reputation Count and Queuing

**Initial rep** = 100 points for everyone.

- Submit a work at deadline daytime: rep −2.
- Submit a work at deadline night: rep −5 .
- Submit a work on the eve of Reset Event: rep−10.
- Artefacts of someone's else code: rep−25.
- Unable to explain self code: rep−50.
- Submit a work on the eve of exam (daytime): rep−35.
- Submit a work on the eve of exam (night): rep−50.
- Submit a work not assigned to self: rep−75.
- Other (violating RoFL Farm rules, etc) — ad hoc.

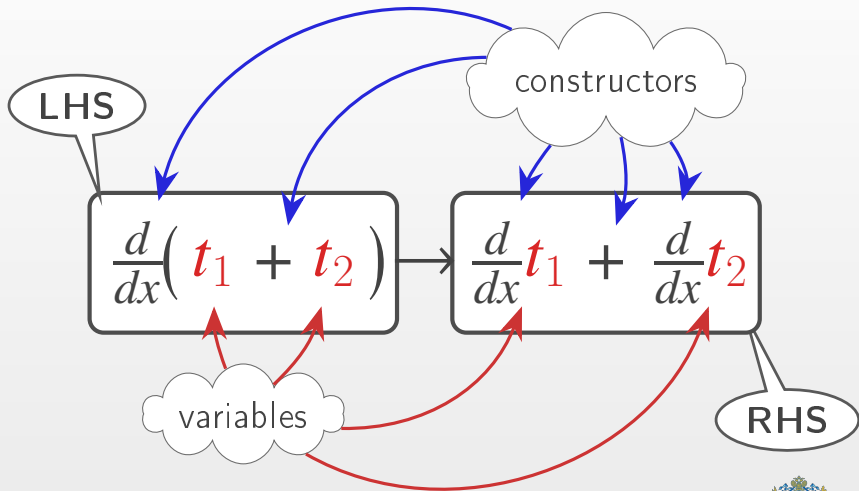
*The assignments are considered in **descending order** of rep!*  
 $\text{rep} \leq 0 \Rightarrow \text{personal task arrangement.}$





# Term Rewriting and All That

Any symbolic computation is a term rewriting process, controlled by an appropriate set of *rewrite rules* — *term rewriting system*.



# Unification and Pattern Matching

A rewriting process *unifies* term  $t$  and LHS  $s_1$  of a rule  $s_1 \rightarrow s_2$  by constructing a substitution.

## ☹☹ Unification Formally

Given two terms  $t_1, t_2$ , a **unifier** is a pair of variable substitutions  $\langle \theta_1, \theta_2 \rangle$  s.t.  $t_1\theta_1 = t_2\theta_2$ .

If  $\text{eq}(x, x) \rightarrow \text{True}$ , when  $\text{eq}\left(\begin{array}{c} \text{A} \\ \text{Y} \end{array}, \begin{array}{c} \text{B} \\ \text{X} \end{array}\right)$  is rewritten?

Unification is possible when  $x = \begin{array}{c} \text{A} \\ \text{B} \end{array} \text{A}$

Term  $\text{eq}\left(\begin{array}{c} \text{A} \\ \text{Y} \end{array}, \begin{array}{c} \text{B} \\ \text{X} \end{array}\right)$  is never unified with  $\text{eq}(x, x)$ .



# Reduction, Redexes, Normal Forms

Given a TRS  $\left\{ \Phi_i \rightarrow \Psi_i \right\}_{i=1}^n$  and a term  $T$ ,

## ☹☹ Basic Notions of Rewriting

- A **redex** is a subterm  $T_0$  that can be unified with some  $\Phi_i$  by unifier  $\langle \theta_1, \theta_2 \rangle$ .
- **Reduction** replaces  $T_0\theta_1$  in  $T$  by  $\Psi_i\theta_2$ .
- Term  $T$  is in a **normal form** if  $T$  contains no redex.
- **Normalisation** — reduction to the normal form:  $T \rightarrow T'$ .

Given TRS  $\begin{cases} 0 + x \rightarrow x \\ 0 \cdot x \rightarrow 0 \end{cases}$ , the term  $(0 + 1) \cdot 0$  contains a redex  $0 + 1$ , and the reduced term  $1 \cdot 0$  is in the normal form.

*Adding commutativity rules results in unrestricted reductions.*



# $\alpha$ -conversions and $\alpha$ -equivalence

- If  $x, z$  are variables, the rewriting rules  $\{\text{eq}(x, x) \rightarrow \text{True}\}$  and  $\{\text{eq}(z, z) \rightarrow \text{True}\}$  are *equivalent*; while the rule  $\{\text{eq}(x, z) \rightarrow \text{True}\}$  is not equivalent to both.
- $\alpha$ -conversion is a semantic-preserving variable renaming. Non-trivial in case of bound variables.

---

## $\lambda$ -calculus

---

- Constructors:  $\text{Apply}(M, N)$  and  $\lambda x.M$ , bounding  $x$  in  $M$ .
- Rewriting rule:  $\text{Apply}(\lambda x.M, N) \rightarrow M[x := N]$ .

---

***Reduction + capture-avoiding substitution = universal computation model.***



# Proving Properties of FL and Term Ordering

*Check that the language  $\mathcal{L} = \{w \mid |w|_a \text{ is even}\}$  is the set of normal forms generated from term srt  $\{S\}$  using the following TRS  $T$ :  $S \rightarrow a S a \quad S \rightarrow b S \quad S \rightarrow S b \quad S \rightarrow \varepsilon$*

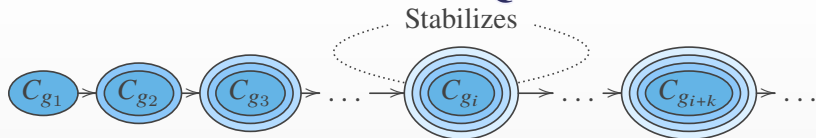
- Check that all words from  $\mathcal{L}$  can be generated by  $T$ .
- Check that all the normal forms  $\eta$  s.t.  $S \twoheadrightarrow \eta$  are in  $\mathcal{L}$ .

Assume that there exists  $w$  s.t.  $S \twoheadrightarrow w$  and  $|w|_a$  is odd. The reduction step before the application of  $\{S \rightarrow \varepsilon\}$  can always be replaced by  $\{S \rightarrow \varepsilon\} \Rightarrow$  making the word  $w$  shorter. We can repeat this process to  $\infty$ , contradicting that  $|w|$  is finite.

***Key moment: the length ordering admits no infinite descending chains.***



# Noetherian Orders and Well-Quasi-Orders



## ☹☹ Ordering as an Induction Base

- A preorder  $\preceq$  is **well-founded** on set  $\mathcal{A}$  if it admits no infinite descending chains, i.e. all chains  $t_0 \succeq t_1 \succeq \dots$  are finite.
- A preorder is **Noetherian** on set  $\mathcal{A}$  if it stabilizes upwards (no infinite strictly ascending chains, abbreviated ACC).
- A preorder  $\preceq$  is a **well-quasi-order** on set  $\mathcal{A}$  if every infinite term sequence from  $\mathcal{A}$  contains  $t_i, t_j$  s.t.  $\left(i < j \ \& \ t_i \preceq t_j\right)$ .



# Well-Founded Monotone Algebras

A preorder  $\preceq$  is monotone on  $\mathcal{A}$ , if

$$\forall f, t_1, \dots, t_n, s, s' \in \mathcal{A} \left( s \preceq s' \Rightarrow f(t_1, \dots, s, \dots, t_n) \preceq f(t_1, \dots, s', \dots, t_n) \right),$$

and is strictly monotone, if converse always fails.

---

## *Well-Founded Monotone Algebras*

---

WFMA over the signature  $F$  over a well-founded set  $\langle \mathcal{A}, > \rangle$  s.t. for any  $f \in F$  there exists a function  $f_{\mathcal{A}} : \mathcal{A}^n \rightarrow \mathcal{A}$  being strictly monotone wrt all its arguments.

---

- $\langle \mathbb{N}, > \rangle$  is a WFMA over a signature  $(f_1, 2), (f_2, 1)$ ;
- $\langle \mathbb{Q}^+, > \rangle$  and  $\langle \mathbb{Z}, > \rangle$  are not WFMA.



# TRS Termination

Given a variable set  $V$ , the extension  $\sigma$  of  $V$  to  $\mathcal{A}$  is defined as follows:

- $[x, \sigma] = \sigma(x)$ ;
- $[f(t_1, \dots, t_n), \sigma] = f_{\mathcal{A}}([t_1, \sigma], \dots, [t_n, \sigma])$ .

A TRS  $\{l_i \rightarrow r_i\}$  is compatible with WFMA  $\mathcal{A} \Leftrightarrow$  for all  $i$ ,  $\sigma$  condition  $[l_i, \sigma] > [r_i, \sigma]$  holds.

## *Main Theorem on TRS Termination*

A TRS  $\{l_i \rightarrow r_i\}$  terminates  $\Leftrightarrow$  there exists a WFMA compatible with  $\{l_i \rightarrow r_i\}$ .

*The TRS  $\left\{ \frac{d}{dx}(t_1 + t_2) \rightarrow \frac{d}{dx}t_1 + \frac{d}{dx}t_2 \right\}$  is terminating, which is verified by WFMA over  $\langle \mathbb{N} \setminus \{0\}, > \rangle$ :*

- $f_A(+ (u, v)) = u + v + 1$ ;
- $f_A(\frac{d}{dx}(u)) = 2 \cdot u$ .





# Free Monoid (aka String DataType) as a FL Carrier

Usual case: the string data type is a carrier of a TRS; the only constructor is the string concatenation.

Then the term rewriting system becomes *a string rewriting system* (SRS):  $\{l_i \rightarrow r_i\}$ , where  $l_i, r_i$  are strings.

## ☹☹ Terminal & Nonterminal Symbols

*Given disjoint alphabets  $N, \Sigma$ , we say that elements of  $\Sigma$  are terminal symbols, and elements of  $N$  are non-terminals in a given SRS  $\{l_i \rightarrow r_i\}$ , if words  $w \in N \cup \Sigma$  containing letters from  $N$  are considered as partially computed (even if they are in the normal form).*



# Formal Grammars

## ☹☹ Definition

A *grammar* is a tuple  $G = \langle N, \Sigma, P, S \rangle$ , where:

- $N$  — non-terminal alphabet;
- $\Sigma$  — terminal alphabet;
- $P$  — string rewriting system  $\{\alpha_i \rightarrow \beta_i\}$ , where  $\alpha_i$  is non-empty;
- $S \in N$  is an initial nonterminal.

A *language*  $\mathcal{L}(G)$  of a grammar  $G$  is the set  $\{u \mid u \in \Sigma^* \text{ \& } S \rightarrow^* u\}$ , where  $\rightarrow^*$  is a composition of reductions.

