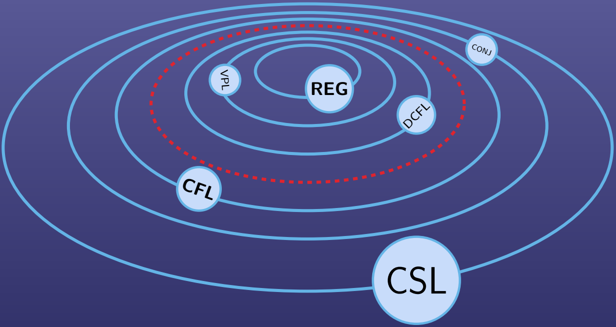
Bauman Moscow State University
Th. Computer Science Dept.

Introduction to Realms of Formal Languages

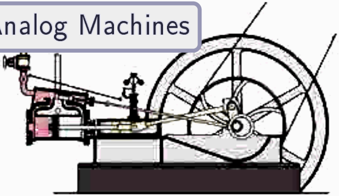


Antonina Nepeivoda
a_nevod@mail.ru

Lecture Outline

Physical Concept of Automaton

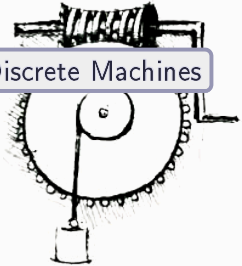
Analog Machines



Continuous control

Differential equations

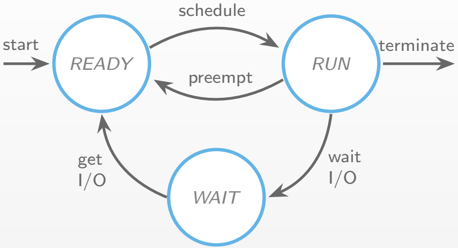
Discrete Machines



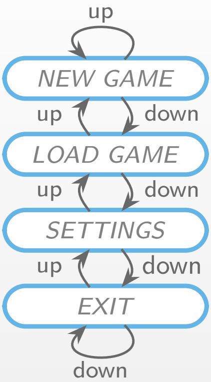
Discrete control

Algebra & Logics

Automata Specifications



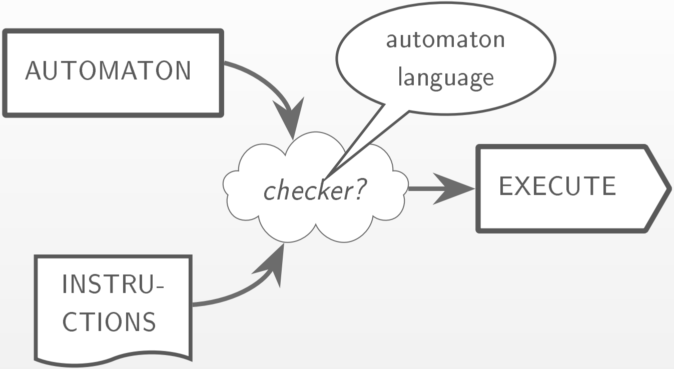
- CPU Scheduler scheme



- Simple game menu

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

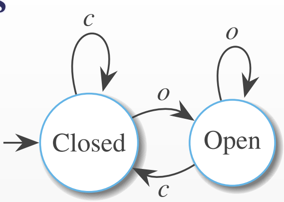
Programming: Automata Languages



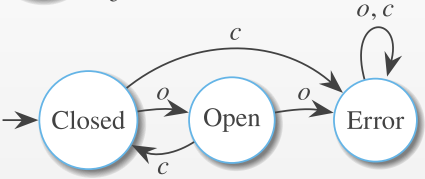
- One-by-one input steps ⇒ automaton model is enough.
- Sequence of multiple steps given simultaneously ⇒ executor is required to verify the sequence wrt the automata model.
- Admissible step sequences — *a formal language* of the machine.

Automata Models

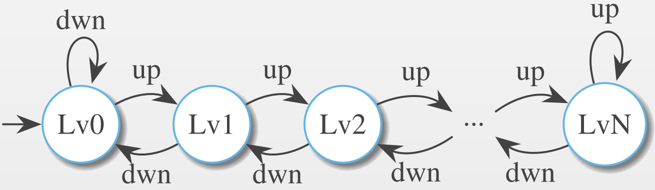
Door automaton:



Channel automaton:



Elevator:



Prehistoric Era of Formalization

Quando chel cubo con le cose appresso
Se aqquaglia a qualche numero discreto
Trouan duo altri di erenti in esso
Dapoi terrai questo per consueto
Che llor productto sempre sia equale
Alterzo cubo delle cose neto,
El residuo poi suo generale
Delli lor lati cubi ben sottrati
Varra la tua cosa principale.
...

Tartaglia, 1539

$3x^2 + 1 - (10x^3 + 2x) = 4$ in Diophantus's notation:

$$\Delta^{\upsilon}\gamma\mathring{M}\alpha \pitchfork K^{\upsilon}\iota\varsigma\beta\iota^{\sigma}\mathring{M}\delta$$

Antique Era of Formal Languages

Von Dyck's Theorem, 1882

Let X be a set and let R be a set of reduced words on X. Assume that a group G has the presentation $\langle X \mid R \rangle$. If H is any group generated by X and H satisfies the relations of G, i.e., $\omega = 1$ in H for all $\omega \in R$, then there is a surjective group homomorphism from G to H.

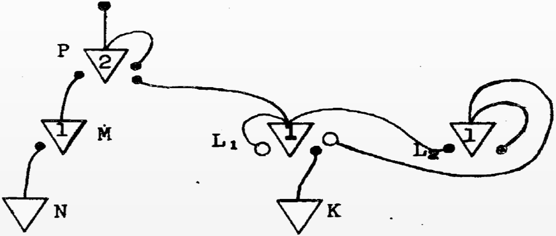
Axel Thue's Theorems on Unavoidable Patterns, 1906

In the alphabet $\{a, b, c, d\}$, there exists an infinite square-free word (i.e. not containing adjacent equal factors, or not matching a pattern $xwwy$ with w non-empty).

Later, in 1912, A. Thue improved his result for the alphabet $\{a, b, c\}$.

Renaissance of Formal Languages

McCulloch& Pitts Neural Networks:

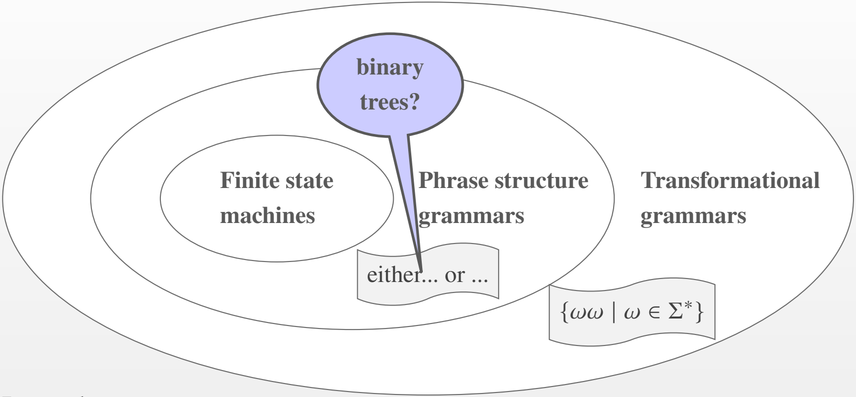


- • — excitatory signal;
- ∘ — inhibitory signal;
- ▽ — an input neuron;
- k (inside ▽) — an inner neuron firing whenever none of the inhibitory signals and at least k of excitatory signals fire.

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

Noam Chomsky and Language Hierarchy

Three Models For The Description of Language, 1956:



In modern terms:

- phrase-structure grammars (1956) ⇒ context-free grammars
- transformational grammars (1956) ⇒ recursively-enumerable grammars

Binary Trees and Von Dyck's Language

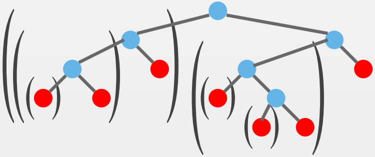
Dyck's Language

One-sided (von) Dyck's language is the language of balanced parentheses.

One is able to code tree structures with strings using the parentheses language.

- One-to-one correspondence between words of length $2 \cdot n$ and binary trees with n inner vertices (using $V_P \to (V_R)V_L$).

$$\Bigg(\bigg(\Big(\Big)\bigg)\Bigg)\Bigg(\Big(\Big)\Big(\Big)\Bigg) \longleftrightarrow$$



Binary Trees and Von Dyck’s Language

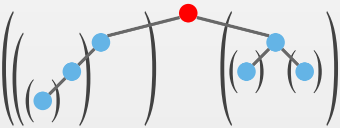
Dyck’s Language

One-sided (von) Dyck’s language is the language of balanced parentheses.

One is able to code tree structures with strings using the parentheses language.

- One-to-one correspondence between words of length $2 \cdot n$ and arbitrary trees with $n + 1$ vertices (DFS).

$$\Bigg(\bigg(\Big(\,\Big)\bigg)\Bigg)\Bigg(\Big(\,\Big)\Big(\,\Big)\Bigg) \longleftrightarrow$$



Binary Trees and Von Dyck's Language

👀 Dyck's Language

One-sided (von) Dyck's language is the language of balanced parentheses.

One is able to code tree structures with strings using the parentheses language.

- One-to-one correspondence between words of length $2 \cdot n$ and binary trees with n inner vertices (using $V_P \longrightarrow (V_R)V_L$).
- One-to-one correspondence between words of length $2 \cdot n$ and arbitrary trees with $n + 1$ vertices (DFS).

Two-sided Dyck's languages are the languages of words in a free group that are reduced to 1.

Text Analysis and Forbidden Patterns

- Russian and English texts naturally avoid the pattern $\overbrace{x^2y^2z^2w^2}^{\substack{xx\,yy\,zz\,ww,\\ x,\,y,\,z,\,w \text{ non-empty}}}$.
- Still, pattern $x^2y^2z^2w^2$ occurs in everyday Chinese language, e.g.:

明明白白清清楚楚

- Any well-composed natural text and programming language source code written by hand should avoid the pattern x^2 (tandem repeats) with $\underbrace{|x|}_{\text{length of } x}$ large enough.
- Still, generated and auto-optimised code can contain this pattern naturally (e.g. in loop unrolling).
- Within DNA, tandem repeats (and even more, multiple repeats x^n with n large enough) are a crucial part of the gene code.

Capturing Invariants with Formal Languages

```
while (x > 0) {
 // some code not involving x and y
 x = x - y;
 }
```

- Minimal formal language validating loop termination: $\{= 0, \neq 0\}$.
- Better: $\{-, 0, +\}$.
- Possibly more precise, also for overflows: intervals.

```
while (x.contains('<script>')) {
 // x = x.replace('<', '< '); // invalid
 // x = x.replaceAll('<', '< '); // valid
 // x = x.replace('<', '&lt;'); // valid
  x = x.replace('<s', '< s'); } // valid
```

- Minimal language validating loop termination: tracking $\overbrace{|x|_{<\mathtt{s}}}^{\text{occurrences of substring } <\mathtt{s}}$.

Formal Languages Formally

Definition

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

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:

- $\{\underbrace{aa\ldots a}_{n \text{ times}}\ \underbrace{bb\ldots b}_{n\cdot 3 \text{ times}} \mid n \in \mathbb{N}\}$ (henceforth $\{a^n b^{3n} \mid n \in \mathbb{N}\}$);
- words containing a square number of letters a

  $$\Big\{w \mid \exists k(\underbrace{|w|_a}_{\text{number of letters } a \text{ in word } w} = k^2)\Big\};$$
- Russian palindromes of the even length

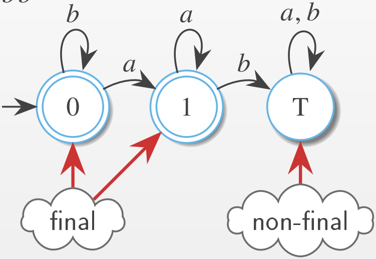
  («*Я слипся, я слился, я спился*»);
- sequences of balanced parentheses;
- well-formed arithmetic expressions over $\mathbb{N}$ and $\{\cdot, +\}$.

semantics:

- all tautologies in the classical logic;
- all consistently typed programs in Python;
- formal languages with linear-time-decidable membership.

Varieties of FL Representations

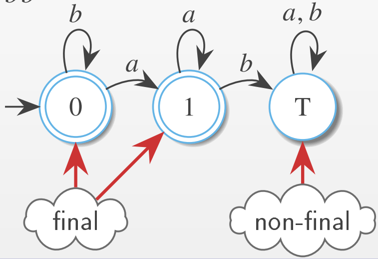
- Set comprehension: $\left\{w \in \{a,b\}^* \mid w \text{ does not include } ab\right\}$, or $\{w \mid |w|_{ab} = 0 \ \&\ w \in \{a,b\}^*\}$.
- Algebraic expressions: $b^* a^*$.
- Term rewriting systems: $\begin{cases} a \to aa \\ b \to bb \end{cases}$, base set $\{\overbrace{\ \varepsilon\ }^{\text{empty word}}, a, b, ba\}$.
- Recognising machines:



- First-order or second-order logical formulas:
$\forall x, y\big(\underbrace{Q_a(x)}_{x \text{ is equal to } a} \ \&\ \underbrace{\mathcal{S}(x,\ y)}_{y \text{ succeeds } x} \ \Rightarrow\ \underbrace{\neg Q_b(y)}_{y \text{ is not equal to } b}\big)$.

Varieties of FL Representations

- Set comprehension: $\{w \in \{a,b\}^* \mid w \text{ does not include } ab\}$, or $\{w \mid |w|_{ab} = 0 \;\&\; w \in \{a,b\}^*\}$.
- Algebraic expressions: $b^* a^*$.
- Term rewriting systems: $\begin{cases} a \to aa \\ b \to bb \end{cases}$, base set $\{\overbrace{\quad\varepsilon\quad}^{\text{empty word}}, a, b, ba\}$.
- Recognising machines:



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

The Empty Word Notation

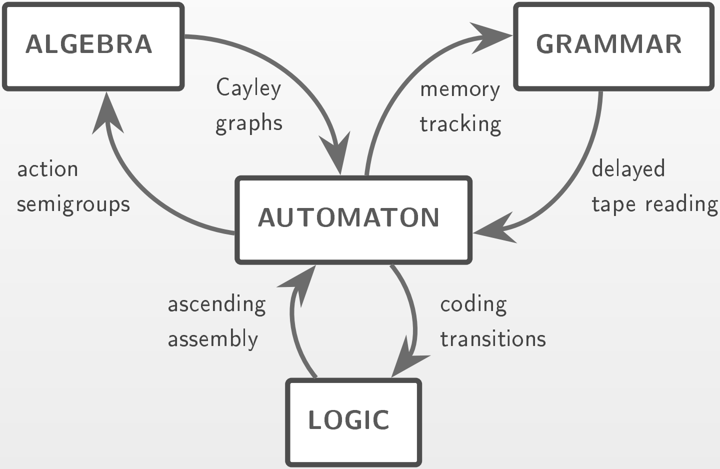
- In modern computer science (2000s and later) — epsilon ε.
- Before 2000s — lambda λ.

The reason: λ-notation for anonymous functions. $\overbrace{\lambda\text{Arg.Body}}^{\text{in theory}}$ or $\overbrace{\lambda\text{Arg} \to \text{Body}}^{\text{in applied research}}$, although known from 1930-s, became widespread and met formal language theory.

Compare: $\lambda\langle q_i, \lambda, q_j\rangle.\lambda$ and $\lambda\langle q_i, \varepsilon, q_j\rangle.\varepsilon$.
Or: $\lambda\big(q_i \xrightarrow{\lambda} q_j\big) \to \lambda$ and $\lambda\big(q_i \xrightarrow{\varepsilon} q_j\big) \to \varepsilon$.

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

- Rewriting and Syntactic Monoids;
- 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;
- Ehrenfeucht–Fraïssé games and Pumping;
- Hardest Languages and Language Representations;
- Language Inference;
- Combinatorial & Computational Language Properties.

- *Midterm II*
- Concluding Lecture
- *«RoFL Farm»*

Course Scores

👀 Score Arrangement

- ***Midterms*** *× 2 × 15+.*
- ***Assignments*** *× 5 × 8+.*
  - *Java, Python, Go, JS — no score bonus*
  - *C/C++, Kotlin, Dart, TypeScript, Lua, Julia — +1 point*
  - *Rust, Lisp dialects, Scala — +2 points*
  - *Haskell, Erlang — +3 points*
  - *Рефал — +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

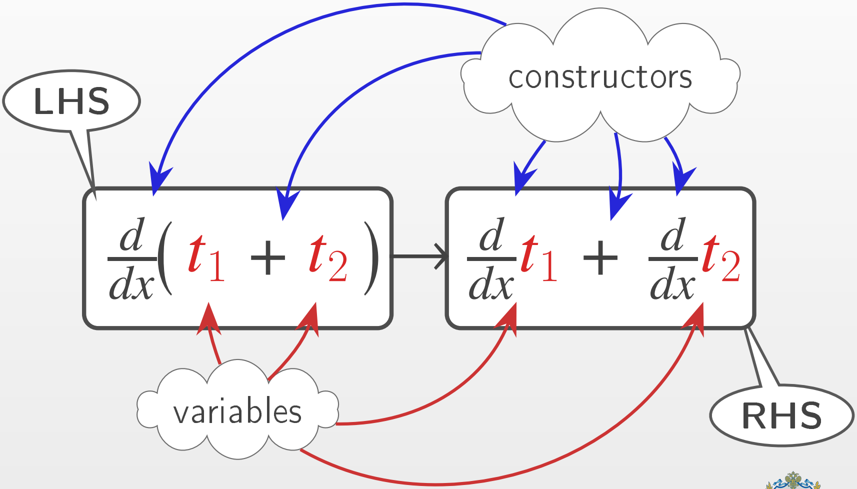
Initial rep = 100 points for everyone.

- 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!*
>
> rep $\leq 0 \Rightarrow$ *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*.



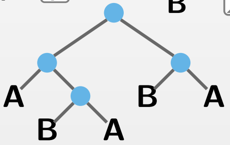
Unification and Pattern Matching

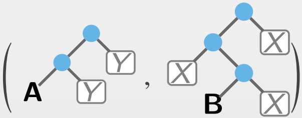
A rewriting process *unifies* term t and LHS s_1 of a rule $s_1 \to s_2$ by constructing a substitution.

👀 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 $\mathrm{Eq}(x, x) \to$True, when Eq  is rewritten?

Unification is possible when $x =$ 

Term Eq  is never unified with $\mathrm{Eq}(x, x)$.

Notation Clash: Substitutions

In algebraic courses, given a substitution σ, its application to a term Φ is denoted with $\sigma(\Phi)$.
In mathematical logic & computer science, a ***postfix notation*** $\Phi\sigma$ is more usual.

Origins of the Notation

In classical mathematical logic textbooks (Tarski, Curry) the substitutions were denoted by $[x/A]$. Hence, the postfix notation was natural: formula $F(x,x)[x/A]$ is pretty more readable than $[x/A](F(x,x))$.

In modern computer science, the notations $[x := A]$ are $[x \mapsto A]$ are both widely used.

Reduction, Redexes, Normal Forms

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

👀 Basic Notions of Rewriting

- ***A redex*** *is a subterm T_0 that can be unified with some Φ_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: $T \twoheadrightarrow T'$.*

Given TRS $\begin{cases} 0 + x \to x \\ 0 \cdot x \to 0 \end{cases}$, 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.

α-conversions and α-equivalence

- If x, z are variables, the rewriting rules $\{\mathrm{Eq}(x, x) \rightarrow \mathrm{True}\}$ and $\{\mathrm{Eq}(z, z) \rightarrow \mathrm{True}\}$ are *equivalent*; while the rule $\{\mathrm{Eq}(x, z) \rightarrow \mathrm{True}\}$ is not equivalent to both.

- α-conversion is a semantic-preserving variable renaming. Non-trivial in case of bound variables.

λ-calculus

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

Reduction + capture-avoiding substitution = universal computation model.

Proving Properties of FL and Term Ordering

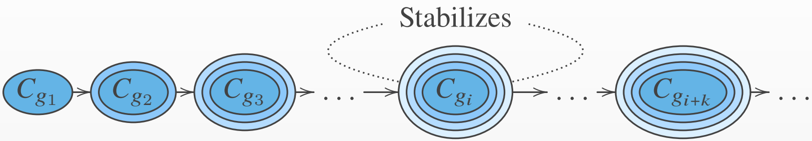
Check that the language $\mathscr{L} = \{w \mid |w|_a \text{ is even}\}$ is the set of normal forms generated from term srt $\{S\}$ using the following TRS $\mathcal{T}$: $\quad S \to a\,S\,a \quad S \to b\,S \quad S \to S\,b \quad S \to \varepsilon$

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

Assume that there exists w s.t. $\overbrace{S \to^* w}^{\text{arbitrary number of steps}}$ and $|w|_a$ is odd. The reduction step before the application of $\{S \to \varepsilon\}$ can always be replaced by $\{S \to \varepsilon\} \Rightarrow$ making the word w shorter. We can repeat this process to ∞, contradicting that $|w|$ is finite.

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

Noetherian Orders and Well-Founded 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).*

Well-Founded Monotone Algebras

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

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

and is strictly monotone, if converse always fails.

Well-Founded Monotone Algebras

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

- $\langle \mathbb{N}, > \rangle$ can be a WFMA over a signature $(f_1, 2), (f_2, 1)$, given e.g. interpretations $f_1(x, y) = x + y + 1$, $f_2(x) = 2 \cdot x$;
- $\langle \mathbb{Q}^+, > \rangle$ and $\langle \mathbb{Z}, > \rangle$ are not WFMAs.

TRS Termination

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

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

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

Informally: the image of l_i is greater than the image of r_i for the extension of any substitution $\sigma : \mathcal{V} \to \mathcal{A}$.

Consider TRS $\{f(f(x)) \to f(x)\}$ and WFMA carrier $\mathcal{N} = \langle \mathbb{N}, > \rangle$.

- *Let $f_{\mathcal{N}}(x) = 2 \cdot x$ and $\sigma = [x := 0]$: $f_{\mathcal{N}}(f_{\mathcal{N}}(x))\sigma = f_{\mathcal{N}}(x)\sigma = 0$, $\Rightarrow$ WFMA is not compatible with the TRS.*
- *Let $f_{\mathcal{N}}(x) = x + 1$: $\forall x \big(f_{\mathcal{N}}(f_{\mathcal{N}}(x)) - f_{\mathcal{N}}(x) = 1\big) \Rightarrow$ the TRS and WFMA are compatible.*

WFMA and TRS

Main Theorem on TRS Termination

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

The TRS $\left\{\frac{d}{dx}(t_1 + t_2) \to \frac{d}{dx}t_1 + \frac{d}{dx}t_2\right\}$ *is terminating, which is verified by WFMA* $\mathcal{N}$ *over* $\langle \mathbb{N}, > \rangle$, $\mathbb{N}$ *not including* 0*:*

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

Indeed, $\frac{d}{dx}_{\mathcal{N}}(t_1 +_{\mathcal{N}} t_2) = 2 \cdot (t_1 + t_2 + 1) = 2 \cdot t_1 + 2 \cdot t_2 + 2$, which is strictly greater than
$\frac{d}{dx}_{\mathcal{N}} t_1 +_{\mathcal{N}} \frac{d}{dx}_{\mathcal{N}} t_2 = 2 \cdot t_1 + 2 \cdot t_2 + 1$.

Choosing a Monotone Interpretation

If $\mathcal{A}$ is well-founded and $f_{\mathcal{A}}$ is strictly monotone, then $\forall a\big(f_{\mathcal{A}}(a) \succeq a\big)$.

Indeed, let $a \succ f_{\mathcal{A}}(a)$. Then the chain

$$a, f_{\mathcal{A}}(a), f_{\mathcal{A}}(f_{\mathcal{A}}(a)), \ldots, f_{\mathcal{A}}^{n}(a), \ldots$$

is infinitely descending.

Good choices of functions over $\mathcal{A}$:

- given $\mathcal{A} \subseteq \mathbb{N}$, extended polynomials strictly increasing on $\mathcal{A}$, e.g. $f(x) = x \cdot \lfloor \log_2 x \rfloor$ on $[4; +\infty)$;
- given $\mathcal{A} \subseteq \mathbb{N}^k$, functions strictly increasing lexicographically on $\mathcal{A}$, e.g. $f(\langle x, y \rangle) = \langle x + 1, 0 \rangle$ on $\mathbb{N}^2$.

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): $\{l_i \to r_i\}$, where l_i, r_i are strings.

👀 Terminal & Nonterminal Symbols

Given disjoint alphabets N, Σ, we say that elements of Σ are terminal symbols, and elements of N are non-terminals in a given SRS $\{l_i \to r_i\}$, 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;*
- *Σ — terminal alphabet;*
- *P — string rewriting system $\{\alpha_i \to \beta_i\}$, where α_i is non-empty;*
- *$S \in N$ is an initial nonterminal.*

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

Diamond Property, or Confluence

Evaluation of an arithmetic expression:

$$\begin{tikzcd}
 & (2+3)\cdot(4+5) \arrow[dl] \arrow[dr] & \\
5\cdot(4+5) \arrow[dr] & & (2+3)\cdot 9 \arrow[dl] \\
 & 45 &
\end{tikzcd}$$

A unique normal form exists.

Non-deterministic replacements (rewriting) in strings:

$$\begin{tikzcd}
 & ababa \arrow[dl, "aba \mapsto b"'] \arrow[dr, "aba \mapsto b"] & \\
bba & & abb
\end{tikzcd}$$

No unique normal form.

Given a TRS $\mathcal{T}$, it is ***confluent*** *iff*

$$\forall t_0, t_1, t_2 \exists t_3 \big(t_0 \xrightarrow{\mathcal{T}}{}^{*} t_1 \;\&\; t_0 \xrightarrow{\mathcal{T}}{}^{*} t_2 \Rightarrow t_1 \xrightarrow{\mathcal{T}}{}^{*} t_3 \;\&\; t_2 \xrightarrow{\mathcal{T}}{}^{*} t_3\big).$$

Global and Local Confluence

- (Global) confluence:

$$\forall t_0, t_1, t_2 \exists t_3 \big(\underbrace{t_0 \xrightarrow{\mathcal{T}}{}^{*}\ t_1}_{\text{arbitrary number of steps}} \ \&\ t_0 \overbrace{\xrightarrow{\mathcal{T}}{}^{*}}^{\text{arbitrary number of steps}} t_2 \Rightarrow t_1 \xrightarrow{\mathcal{T}}{}^{*}\ t_3 \ \&\ t_2 \xrightarrow{\mathcal{T}}{}^{*}\ t_3\big).$$

- Local confluence:

$$\forall t_0, t_1, t_2 \exists t_3 \big(\underbrace{t_0 \xrightarrow{\mathcal{T}} t_1}_{\text{one-step rewriting}} \ \&\ t_0 \overbrace{\xrightarrow{\mathcal{T}}}^{\text{one-step rewriting}} t_2 \Rightarrow t_1 \xrightarrow{\mathcal{T}}{}^{*}\ t_3 \ \&\ t_2 \xrightarrow{\mathcal{T}}{}^{*}\ t_3\big).$$

The following TRS is locally confluent, but not confluent:

$$\infty \to \infty + 1 \quad \mathrm{Eq}(x,\ x) \to \mathrm{True} \quad \mathrm{Eq}(x,\ x+1) \to \mathrm{False}$$

Global and Local Confluence

- (Global) confluence:

$$\forall t_0, t_1, t_2 \exists t_3 \big(\underbrace{t_0 \xrightarrow{\mathcal{T}}{}^{*} t_1}_{\text{arbitrary number of steps}} \;\&\; t_0 \overbrace{\xrightarrow{\mathcal{T}}{}^{*}}^{\text{arbitrary number of steps}} t_2 \Rightarrow t_1 \xrightarrow{\mathcal{T}}{}^{*} t_3 \;\&\; t_2 \xrightarrow{\mathcal{T}}{}^{*} t_3\big).$$

- Local confluence:

$$\forall t_0, t_1, t_2 \exists t_3 \big(\underbrace{t_0 \xrightarrow{\mathcal{T}} t_1}_{\text{one-step rewriting}} \;\&\; t_0 \overbrace{\xrightarrow{\mathcal{T}}}^{\text{one-step rewriting}} t_2 \Rightarrow t_1 \xrightarrow{\mathcal{T}}{}^{*} t_3 \;\&\; t_2 \xrightarrow{\mathcal{T}}{}^{*} t_3\big).$$

👀 **Newman's Lemma**

If TRS is terminating, then local and global confluence properties coincide.