Turchin's Theorem in Formal Language Theory

Antonina Nepeivoda
a_nevod@mail.ru

Formal Language Analysis

Regular

- Myhill–Nerode characterisation;
- Generic Pumping Lemma;
- etc (finite monoids, MSO)

Context-Free

- Lots of Pumping Lemmas (Classical, Ogden, Bader–Moura...)
- counting arguments (Parikh images...)

Context-Sensitive

- Complexity arguments
- Combinatorics
- Class-specific lemmas

Classical Pumping Lemma

Let $\mathscr{L}$ be a CFL. Then there exists a pumping length $p \in \mathbb{N}$ s.t. for every $w \in \mathscr{L}$, $|w| \leq p$ the following condition holds:

$$\exists x_i, y_i, z\big(w = x_1 y_1 z\, y_2 x_2 \ \&\ |y_1 y_2| \geq 1 \ \&\ |y_1 z y_2| \leq p \\ \&\ \forall k \in \mathbb{N}(x_1 y_1^k z y_2^k x_2 \in \mathscr{L})\big)$$

- Forced pumping positions (Ogden's lemma);
- Forbidden pumping positions (Bader–Moura theorem);
- Multiple pumping (Multiple Pumping Lemma)...

Greibach Normal Form & Stack

Let $\mathscr{L}$ be a CFL. Then it can be generated by a grammar G whose rules are of the forms $N_i \mapsto \gamma_i$ and $N_i \mapsto \gamma_i M_{1,i} \dots M_{k,i}$, where $\gamma_i \in \Sigma$, $N_i, M_j \in \mathcal{N}$

Parsing by such a G can be considered as a computation over the set of null-ary functions, represented by non-terminals (elements of N). Thus, the part $M_{1,i} \dots M_{k,i}$ represents the stack change on the pair $\langle N_i, \gamma_i \rangle$.

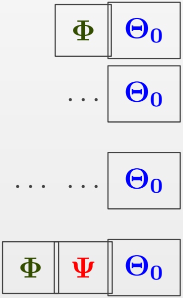
Greibach Normal Form & Stack

Let $\mathscr{L}$ be a CFL. Then it can be defined by a program P whose rules $N_i(\gamma_i) \mapsto \varepsilon$ and $N_i(\gamma_i) \mapsto M_{1,i} \dots M_{k,i}$, where $\gamma_i \in \Sigma$, $N_i, M_j \in \mathcal{N}$

- The call-stack behaviour on runs of P is determined by an alphabetic prefix grammar
- (Alphabetic) prefix grammars are known to define regular languages
- ...satisfying pleasant lemmas s.t. generic pumping lemma and Myhill–Nerode properties.

Turchin's Theorem

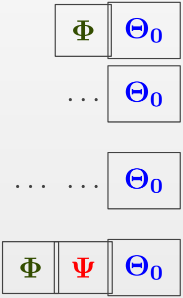
Given a run containing the stack states: $\rho_1 : \Phi\Theta_0$, $\rho_2 : \Phi\Psi\Theta_0$, we say ρ_1 and ρ_2 form a Turchin pair (denoted $\rho_1 \preceq \rho_2$), if Θ_0 does not change on a computation segment starting in ρ_1 and ending in ρ_2.



If Φ comes into an infinite loop generating stack states $\Phi\Psi^n\Theta_0$, then $\rho_1 \preceq \rho_2$. If all the functions are nullary, then the condition $\rho_1 \preceq \rho_2$ is necessary and sufficient to lead to an infinite computation loop.

Turchin's Theorem

Given a run containing the stack states: $\rho_1 : \Phi\Theta_0$, $\rho_2 : \Phi\Psi\Theta_0$, we say ρ_1 and ρ_2 form a Turchin pair (denoted $\rho_1 \preceq \rho_2$), if Θ_0 does not change on a computation segment starting in ρ_1 and ending in ρ_2.

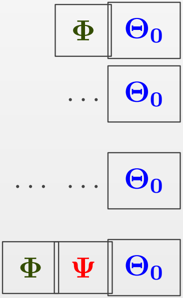


A bad sequence w.r.t. a relation $\preceq$: a segment of a computation not containing pairs in $\preceq$.

Length of a bad sequence w.r.t. $\preceq$ is bounded by the number of rules in program P.

Turchin's Theorem

Given a run containing the stack states: $\rho_1 : \Phi\Theta_0$, $\rho_2 : \Phi\Psi\Theta_0$, we say ρ_1 and ρ_2 form a Turchin pair (denoted $\rho_1 \preceq \rho_2$), if Θ_0 does not change on a computation segment starting in ρ_1 and ending in ρ_2.



Length of a bad sequence w.r.t. $\preceq$ is bounded by the number of rules in program P.

Holds for the stronger case $|\Phi| = 1$.

Turchin's Theorem as a Pumping Lemma

$$
\begin{array}{l}
S \\
\left.\begin{array}{r} \dots \\ A\Theta_0 \end{array}\right\} x_1 \\
\left.\begin{array}{r} \dots \\ A\Psi\Theta_0 \end{array}\right\} y_1 \\
\left.\begin{array}{r} \dots \\ \Psi\Theta_0 \end{array}\right\} z \\
\left.\begin{array}{r} \dots \\ \Theta_0 \end{array}\right\} y_2 \\
\left.\begin{array}{r} \dots \\ \varepsilon \end{array}\right\} x_2
\end{array}
$$

- Can choose any computation segment to be a shortest one;
- Can choose arbitrary finite number of forbidden positions;
- Can choose any long enough computation segment to be pumped;
- Can reason recursively on any computation segment.

Specialization Examples

- Choose the last Turchin's pair on the path and apply the constraints on the Turchin relation bad sequence length:

Let $\mathscr{L}$ be a CFL. Then there exists a pumping length $p \in \mathbb{N}$ s.t. for every $w \in \mathscr{L}$, $|w| \leq p$ the following condition holds:

$$\exists x_i, y_i, z\big(w = x_1 y_1 z\, y_2 x_2 \;\&\; |y_1| \geq 1 \;\&\; |y_2| \geq 1 \;\&\; |z| \geq 1 \\ \;\&\; |y_1 z y_2| \leq p \;\&\; \frac{|x_2|}{|x_1|} \leq p \;\&\; \forall k \in \mathbb{N}(x_1 y_1^k z y_2^k x_2 \in \mathscr{L})\big)$$

Specialization Examples

- Choose the first Turchin's pair on the path and apply the constraints on the Turchin relation bad sequence length:

Let $\mathscr{L}$ be a CFL. Then there exists a pumping length $p \in \mathbb{N}$ s.t. for every $w \in \mathscr{L}$, $|w| \leq p$ the following condition holds:

$$\exists x_i, y_i, z\big(w = x_1 y_1 z\, y_2 x_2 \;\&\; |y_1| \geq 1 \;\&\; |y_2| \geq 1 \;\&\; |z| \geq 1 \;\&\; |x_1 y_1| \leq p \;\&\; \forall k \in \mathbb{N}(x_1 y_1^k z y_2^k x_2 \in \mathscr{L})\big)$$

Specialization Examples

- Reason recursively on computation segments:

Let $\mathscr{L}$ be a CFL. Then there exists a pumping length $p \in \mathbb{N}$ s.t. for every $w \in \mathscr{L}$, $|w| \leq p$ the following condition holds:

$$\exists x_i, y_i, z\big(w = x_1 y_1 z\, y_2 x_2 \;\&\; |y_1| \geq 1 \;\&\; |y_2| \geq 1 \;\&\; |z| \geq 1 \;\&\; \forall k \in \mathbb{N}(x_1 y_1^k z y_2^k x_2 \in \mathscr{L}) \;\&\; (|\xi| \leq p \vee \xi \text{ contains an independent pumping part})\big)$$

- There $\xi \in \{x_1, x_2, z, y_1, y_2\}$.

Discussion

- TT is rather a flexible framework of generating PL instances than a stand-alone theorem;
- TT can be possibly used for some subclasses of context-sensitive languages to construct the approximation lemmas (e.g. for MFA languages: a TT approximation allows us to prove that $a^n b^n$ is not a generic MFA language).
- (hypo!) TT instances are more easily verifiable than CF-PL instances (due to regularity of underlying path language).