An Introduction to Algorithmic Prefix Complexity

Xuangui Huang

Shanghai Jiao Tong University

April 1, 2014
Outline

1 Motivation

2 Partial Recursive Prefix Functions

3 The Invariance Theorem
   - The Theorem
   - Proof Motivation
   - The Construction
   - Proof

4 Examples
We have seen the ‘Algorithmic Plain Complexity’ last week. However,

- it would be more pleasing if the complexity of $xy$ were never less than that of $x$;
- it is natural to restrict effective descriptions to prefix-codes.
We have seen the ‘Algorithmic Plain Complexity’ last week. However,

- it would be more pleasing if the complexity of $xy$ were never less than that of $x$;
- it is natural to restrict effective descriptions to prefix-codes.

Hence, ‘Algorithmic Prefix Complexity’!
We have seen the ‘Algorithmic Plain Complexity’ last week. However,

- it is natural to restrict effective descriptions to prefix-codes.

Hence, ‘Algorithmic Prefix Complexity’!
In plain complexity, we use ‘partial recursive function’. In prefix complexity, we use the following one:
In plain complexity, we use ‘partial recursive function’. In prefix complexity, we use the following one:

**Definition (3.1.1)**

A *partial recursive prefix function* \( \phi: \{0, 1\}^* \to \mathbb{N} \) is a partial recursive (p. r.) function s.t.

- if \( p \) is a proper prefix of \( q \), then at most one of \( \phi(p) \) and \( \phi(q) \) is defined.
Thus we have the following ‘prefix’ version of complexity:

**Definition (‘prefix’ version of 2.1.1)**

Any partial recursive prefix function $\psi$, together with $p$ and $y$, s.t. $\psi(\langle y, p \rangle) = x$, is a description of $x$. The Complexity $C_\psi$ of $x$ conditional to $y$ is defined by

$$C_\psi(x | y) = \min\{l(p) : \psi(\langle y, p \rangle) = x\},$$

and $C_\psi(x | y) = \infty$ if no such $p$ exists.
The Invariance Theorem

Theorem (3.1.1)

There exists an additively optimal p. r. prefix function \( \psi_0 \) s.t.
\[
\forall \text{ p. r. prefix function } \psi, \exists \text{ constant } c_\psi, \text{ s.t.}
\]
\[
\forall x, y \in \mathcal{N}, \ C_{\psi_0}(x|y) \leq C_{\psi}(x|y) + c_\psi.
\]
Similarly as before, we need to construct a universal p.r. prefix function, or, an effective enumeration of p.r. prefix functions.
Similarly as before, we need to construct a universal p.r. prefix function, or, an effective enumeration of p.r. prefix functions.

Fix an effective enumeration of Turing Machines $T_1, T_2, \ldots$, we will effectively construct *prefix machines* $T'_1, T'_2, \ldots$, which will computes all and only the p.r. prefix functions.
Property of our construction

- For Turing Machine $T$, the prefix machine $T'$ we constructed will read the input from left to right in one direction.
Property of our construction

- For Turing Machine $T$, the prefix machine $T'$ we constructed will read the input from left to right in one direction.
- Besides, its input is a potentially infinite binary input sequence $b_1 b_2 \ldots$. 
Construction of $T'$ given $T$

The computation of $T'$ on input $b_1b_2\ldots$ is as follow:

**Step 1** Set $p \leftarrow \epsilon$.

**Step 2** Dovetailing:

- Let $q_j$ is the $j$-th string of $\{0, 1\}^*$
- Execute the following commands stage by stage:
  - In stage $i$, simulate one step of the computation of $T(pq_j)$ for all $j \leq i$
  - If in the above simulation, there exists $T(pq_j) \downarrow$, choose the first $q_j$ as $q$, go to step 3

**Step 3**

- If $q = \epsilon$ Then output $T(p)$ and halt
- Else:
  - read $b \leftarrow$ next input bit, set $p \leftarrow pb$
  - go to step 2
Halting Inputs, or Programs

Definition

For an input sequence \(b_1b_2\ldots\), the halting input, or program, of \(T\) is its initial segment \(b_1b_2\ldots b_m\) s.t. \(T\) halts after reading \(b_m\) before reading \(b_{m+1}\).
Halting Inputs, or Programs

**Definition**

*For an input sequence* $b_1b_2\ldots$, *the halting input, or program, of* $T'$ *is its initial segment* $b_1b_2\ldots b_m$ *s.t. $T'$ halts after reading* $b_m$ *before reading* $b_{m+1}$.

**Fact**

*For a fixed input sequence* $b_1b_2\ldots$, *the program of* $T'$ *is unique.*
Effective Enumeration of p. r. Prefix Functions

- Hence, the programs of $T_1', T_2', \ldots$ induces an effective enumeration of p. r. prefix functions.
Effective Enumeration of p. r. Prefix Functions

- Hence, the programs of $T'_{1}$, $T'_{2}$, ... induces an effective enumeration of p. r. prefix functions.
- More precisely, with slight modification of the construction of $T'$, we can have machine $T''$ read finite input and compute exactly p. r. prefix function.
Proof

Theorem (3.1.1)

There exists an additively optimal p. r. prefix function $\psi_0$ s.t.
\[ \forall \ p. \ r. \ prefix \ function \ \psi, \ \exists \ constant \ c_{\psi}, \ s.t. \]
\[ \forall x, y \in \mathcal{N}, \ C_{\psi_0}(x|y) \leq C_{\psi}(x|y) + c_{\psi}. \]
**Theorem (3.1.1)**

*There exists an additively optimal p. r. prefix function* \( \psi_0 \) *s.t.*

\[
\forall \text{p. r. prefix function } \psi, \exists \text{ constant } c_\psi, \text{ s.t. } \\
\forall x, y \in \mathcal{N}, \ C_{\psi_0}(x|y) \leq C_\psi(x|y) + c_\psi.
\]

**Proof:**

Let \( \psi_0 \) be the p. r. prefix function computed by a universal prefix machine \( U \) s.t.

\[
\forall y, p \in \mathcal{N}, U(\langle y, \langle n, p \rangle \rangle) = T''_n(\langle y, p \rangle), 
\]

Then \( C_{\psi_0}(x|y) \leq C_{\psi_n}(x|y) + (2l(n) + 1) \)
Complexity

Since \( \forall \) additively optimal p. r. prefix functions \( \psi, \psi' \), we have
\[
|C_\psi(x|y) - C_{\psi'}(x|y)| \leq c_{\psi,\psi'},
\]
we can fix one of them as the standard reference \( \psi_0 \).
Complexity

Since ∀ additively optimal p. r. prefix functions ψ, ψ', we have

\[ |C_\psi(x|y) - C_{\psi'}(x|y)| \leq c_{\psi,\psi'}, \]

we can fix one of them as the standard reference ψ₀.

**Definition**

*The prefix complexity of x conditional to y is*

\[ K(x|y) = C_{\psi_0}(x|y). \]

*The (unconditional) prefix complexity of x is*

\[ K(x) = K(x|\epsilon). \]
Advantage of Just Decoding Prefix-codes

- Define $K(x, y) = K(\langle x, y \rangle)$
- Now we can directly concatenate two descriptions to describe $\langle x, y \rangle$. 

Xuangui Huang
An Introduction to Algorithmic Prefix Complexity
Define $K(x, y) = K(\langle x, y \rangle)$

Now we can directly concatenate two descriptions to describe $\langle x, y \rangle$, therefore $K$ is *subadditive*:

$$K(x, y) \leq K(x) + K(y) + O(1)$$
Define $K(x, y) = K(\langle x, y \rangle)$

Now we can directly concatenate two descriptions to describe $\langle x, y \rangle$, therefore $K$ is subadditive:

$$K(x, y) \leq K(x) + K(y) + O(1)$$

Similarly, $K(xy) \leq K(x) + K(y) + O(1)$
Examples

\[ \forall x, K(x) \leq C(x) + K(C(x)) + O(1) \]
\[ \forall x, K(x) \leq C(x) + K(C(x)) + O(1) \]

Proof:
If \( p \) is a shortest program (on TM) for \( x \) with \( l(p) = C(x) \), and \( q \) is a shortest program (on prefix machines) for \( l(p) \) with \( l(q) = K(l(p)) \), then \( qp \) is a program for \( x \) on some prefix machine. \qed
Examples

- $\forall x, K(x) \leq C(x) + K(C(x)) + O(1)$
- $\forall x, y, C(x|y) \leq K(x|y) \leq C(x|y) + 2 \log C(x|y)$
Examples

- \( \forall x, K(x) \leq C(x) + K(C(x)) + O(1) \)
- \( \forall x, y, C(x|y) \leq K(x|y) \leq C(x|y) + 2 \log C(x|y) \)

Proof:
If \( p \) is a shortest program (on TM) for \( x \) conditional to \( y \) with \( l(p) = C(x|y) \),
then \( l(p)p \) is a program for \( p \) on some prefix machine.
Examples

- $\forall x, K(x) \leq C(x) + K(C(x)) + O(1)$
- $\forall x, y, C(x|y) \leq K(x|y) \leq C(x|y) + 2 \log C(x|y)$
- $K(x) \leq K(x|l(x)) + K(l(x)) + O(1) \leq K(x|l(x)) + \log^* l(x) + l(l(x)) + l(l(l(x)))) + \ldots + O(1)$
Thank you!