A Note on Blum Static Complexity Measures

Cezar Câmpeanu*

22.02-2012

*Department of Computer Science and Information Technology
University of Prince Edward Island
CANADA

Blum's Axioms

Blum's Papers on Complexity

- Blum, M.: A machine-independent theory of the complexity of recursive functions. *Journal of the ACM* **14(2)** (1967) 322 – 336
  - It is the foundation for the Theory of Computational Complexity
- Blum, M.: On the size of machines, *Information and Control* **11** (1967) 257–265
  - The starting point of my approach

Blum's Papers on Complexity

- Blum, M.: A machine-independent theory of the complexity of recursive functions. *Journal of the ACM* **14(2)** (1967) 322 – 336
  - It is the foundation for the Theory of Computational Complexity
- Blum, M.: On the size of machines, *Information and Control* **11** (1967) 257–265
  - The starting point of my approach

Blum's Papers on Complexity

- Blum, M.: A machine-independent theory of the complexity of recursive functions. *Journal of the ACM* **14(2)** (1967) 322 – 336
  - It is the foundation for the Theory of Computational Complexity
- Blum, M.: On the size of machines, *Information and Control* **11** (1967) 257–265
  - The starting point of my approach

The Axioms

- "$|\cdot|:\mathbb{N}\longrightarrow\mathbb{N}$ is called a measure of the size of machines, $|i|$ being called the size of M_i, if and only if:
  - there exist at most a finite number of machines of any given size and
  - there exists an affective procedure for deciding, for any y, which machines are of size y."[2]
- "These are all so fantastically weak that any reasonable model of a computer and any reasonable definition of size and step satisfies them" [2]

The Axioms

- "$|\cdot|:\mathbb{N}\longrightarrow\mathbb{N}$ is called a measure of the size of machines, $|i|$ being called the size of M_i, if and only if:
  - there exist at most a finite number of machines of any given size and
  - there exists an affective procedure for deciding, for any y, which machines are of size y."[2]
- "These are all so fantastically weak that any reasonable model of a computer and any reasonable definition of size and step satisfies them" [2]

Plain and Prefix-free Complexity

Known Facts

- Many results for these complexities are exactly the same
  - Many proofs are almost identical
  - Some proofs must take into consideration the fact that the input must have a particular form.
  - There are instances where we have to produce new proofs for each of the two complexities.
- Some results hold for the prefix-free complexity, and do not for plain complexity.
  - Example: the case of infinite sequences.
- WHY?

Known Facts

- Many results for these complexities are exactly the same
  - Many proofs are almost identical
  - Some proofs must take into consideration the fact that the input must have a particular form.
  - There are instances where we have to produce new proofs for each of the two complexities.
- Some results hold for the prefix-free complexity, and do not for plain complexity.
  - Example: the case of infinite sequences.
- WHY?

Known Facts

- Many results for these complexities are exactly the same
  - Many proofs are almost identical
  - Some proofs must take into consideration the fact that the input must have a particular form.
  - There are instances where we have to produce new proofs for each of the two complexities.
- Some results hold for the prefix-free complexity, and do not for plain complexity.
  - Example: the case of infinite sequences.
- WHY?

Unified Approach

Burgin's Work

- Generalized Kolmogorov complexity and other dual complexity measures, Translated from *Kibernetica* **4** (1990) 21–29. Original article submitted June 19 (1986)
- Algorithmic complexity of recursive and inductive algorithms, *Theoretical Computer Science* **317** (2004) 31–60
- Algorithmic complexity as a criterion of unsolvability, *Theoretical Computer Science* **383** (2007) 244–259

Burgin's Conclusion on Randomness

☛ "…an attempt to define in this setting an appropriate concept of randomness was unsuccessful. It turned out that the original definition of Kolmogorov complexity was not relevant for that goal. To get a correct definition of a random infinite sequence, it was necessary to restrict the class of utilized algorithms."

The Axioms Considered by Burgin

$\mathcal{G} \subseteq \mathcal{F}$, $\mathcal{G} = (\phi_i^{(n)})_{i \in I}$ is a class of algorithms.
A (direct) complexity measure is a function $m : I \longrightarrow \mathbb{N}$ such that:

1. (Computational axiom) m is computable;
2. (Re-computational Axiom) the set $\{j \mid m(j) = n\}$ is computable;
3. (Cofinitness Axiom) $\#\{j \mid m(j) = n\} < \infty$.
4. (Re-constructibility Axiom) For any number n, it is possible to build all algorithms A from $\mathcal{G}$ for which $m(A) = n$.
5. (Compositional Axiom) If $A \subseteq B$, then $m(A) \leq m(B)$.

A Follow-up of Burgin's Work

BSC Space

- I only consider axioms 1–4.
- A space $(\mathcal{G}, m)$ satisfying axioms 1–4 is called ***B**lum **S**tatic **C**omplexity space*.
- U is d-universal for $\mathcal{G} = (\psi_i(n))_{i,n}$ if U can emulate any algorithm in $\psi_i(n) = U(d(i), n)$.

Dual Complexity Measures

- Given a complexity measure $m : I \longrightarrow \mathbb{N}$ and $\psi \in \mathcal{G}$, the dual to m with respect to ϕ is

$$m^0_\psi(x) = \min\{m(y) \mid y \in I, \psi(y) = x\}.$$

- If indexes y are over an alphabet with p letters A_p, we may consider as $string^{-1}(y) \in \mathbb{N}$ instead of $y \in A_p^*$, because $string(n)$ is a one to one function.
- The length function on A_p^*, $m(y) = |y|$ induces the dual to length complexity measure.
- Plain and prefix-free complexity measures are dual to length complexity measures. [3]

Complexity of a Number/String

- $$C^{\mathcal{G}}(x) = \inf_{i \in I, y \in I} \{m(i) + m(y) \mid \psi_i(y) = x\}. \quad (1)$$

- $$C^{\mathcal{G}}_{\psi}(x) = \inf_{y \in I} \{m(y) \mid \psi(y) = x\}. \quad (2)$$

- In case ψ is an universal algorithm for $\mathcal{G}$ ($\psi \in \mathcal{G}$), then

$$C(x) = C_{\psi}(x) + O(1).$$

Total Defined Functions

String Functions

- $\mathcal{T} = ((\tau_i)_{i \in I})$ is a set of total functions over A_p^*, and $< \cdot, \cdot >$ is a pairing function
- **Theorem 1.** *If*

  1. *for every $\tau \in \mathcal{T}$, there exists a function B_τ such that*

  $$|\tau(xy)| \leq |\tau(x)| + B_\tau(|y|),$$

  2. *for every $M > 0$, there is $i \in I$ and x such that $|\tau_i(x)| > |x| \cdot M$,*

  then there is no universal function for $\mathcal{T}$ in $\mathcal{T}$.

Examples/Remarks

- Functions that are realized by functional transducers satisfy the above theorem.

  Theorem 9 in [7] is a corollary of Theorem 1.

- The proof of Theorem 1 do not require that all functions of family $\mathcal{T}$ to be total functions, however, we assume that if $\tau(xy)$ is defined, then $\tau(x)$ is also defined.

Computable Complexities

- For any $\tau_i \in \mathcal{T}$, we have that:

$$C(x) \leq C_{\tau_i}(x) + m(i).$$

- If identity function can be encoded by $\mathcal{T}$, then the complexity C is computable.

BUSC Spaces

Encoding

- E is a computable function such that

  1. E is injective and is a length increasing function in the second argument, i.e., there exists c_e, such that if $|x| \leq |y|$, then $|E(i,x)| \leq |E(i,y)| + c_e$.
  2. $|E(i,x)| \leq |E(i',x)| + \eta(i,i')$, for some function $\eta : \mathbb{N}^2 \longrightarrow \mathbb{N}$.

Encoding Families of Functions

- E and e are computable functions, E as above.
- **Definition 4.** *We say that the family $\mathcal{G} = (\psi_j)_{j \in J}$ is an (e, E)-encoding of the family $\mathcal{H} = (\mu_i)_{i \in I}$, if for every $i \in I$ and all $x \in \mathbb{N}$, we have that:*

  1. $\mu_i(x) = \psi_{e(i)}(E(i, x))$, *for all* $i \in I$ *and* $x \in \mathbb{N}$,
  2. *if* $\psi_j(z) = x$, *then* $e(i) = j$ *and* $E(i, y) = z$, *for some* $i \in I$ *and* $y \in \mathbb{N}$.

- A ***B**lum **U**niversal **S**tatic **C**omplexity space* is a BSC space with an universal algorithm.

Results for Encodings of BUSC Spaces

- If $\mathcal{H}$ has an universal algorithm in $\mathcal{H}$, and $\mathcal{G}$ is an encoding of $\mathcal{H}$, then $\mathcal{G}$ has an universal algorithm in $\mathcal{G}$.
- The set $\mathcal{C}_t = \{x \in A_p^* \mid C^{\mathcal{G}}(x) \geq m(x) - t\}$ is immune.
- The function $C^{\mathcal{G}}$ is not computable.
- The set of canonical programs CP is immune.
- The function $f(x) = x^*$ is not computable.
- The set $RAND_t^{\mathcal{G}}$ is immune.

What's Next

Some Results on Finite Transducer Complexity

- All strings with complexity less than 25 (using S_0 encoding defined in [7]) are now known.
- Computing complexity for a string of length l is almost as expensive as computing complexity for all strings with complexity less than $l - 8$.
- As expected, strings like 1^n and 0^n have the lowest complexity.

Results on Finite Transducer Complexity ...

- We could find strings of maximum complexity ($l+8$) for all values of the length l, $3 \leq l \leq 17$.

- There is no string of maximum complexity 8 or 10. Let us define the set of magic number to be

$$Magic = \{m \mid \Sigma^{\mathcal{T}}(n) < n + 8 = m, n \in \mathbb{N}\}. \quad (3)$$

or

$$Magic = \{m \mid m \neq \Sigma^{\mathcal{T}}(n), n \in \mathbb{N}\}. \quad (4)$$

Is there any magic number $m > 17$?

Results on Finite Transducer Complexity ...

- Minimum complexity increases much slower than linear. For example:
  1. the complexity of letters is 7
  2. the minimum complexity of strings of length 10 is 13
  3. the minimum complexity of strings of length 13 is 17
  4. the minimum complexity of strings of length 16 is 16
  5. the minimum complexity of strings of length 17 is 19
- We could find 16 words of length 32 with complexity 16
- We could find 1388 words of length 34 with complexity 24
- We could find 16 words of length 36 with complexity 21 and 32 strings of length 55 with complexity 25.

Future Work (in Progress) and Open Problems

- If two encodings of a BUSC differ by a constant, then the complexities "are equivalent".
- For two encodings of a BUSC where the size of one of them increases "much faster" than the other one, we have a strict inclusion for the set of random strings.
- Define randomness of infinite strings for an arbitrary BUSC space.
- Does it make sense to define randomness for an arbitrary BSC space?
- Give other conditions for encodings, such that the "known" results can still be proved.
- Is there a necessary condition for the (un)computability of a complexity measure?

References

[1] Blum, M.: A machine-independent theory of the complexity of recursive functions, *Journal of the ACM* **14(2)** (1967) 322 – 336

[2] Blum, M.: On the size of machines, *Information and Control* **11** (1967) 257–265

[3] Burgin, M.: Generalized Kolmogorov complexity and other dual complexity measures, Translated from *Kibernetica* **4** (1990) 21–29. Original article submitted June 19 (1986)

[4] Burgin, M.: Algorithmic complexity of recursive and inductive algorithms, *Theoretical Computer Science* **317** (2004) 31–60

References

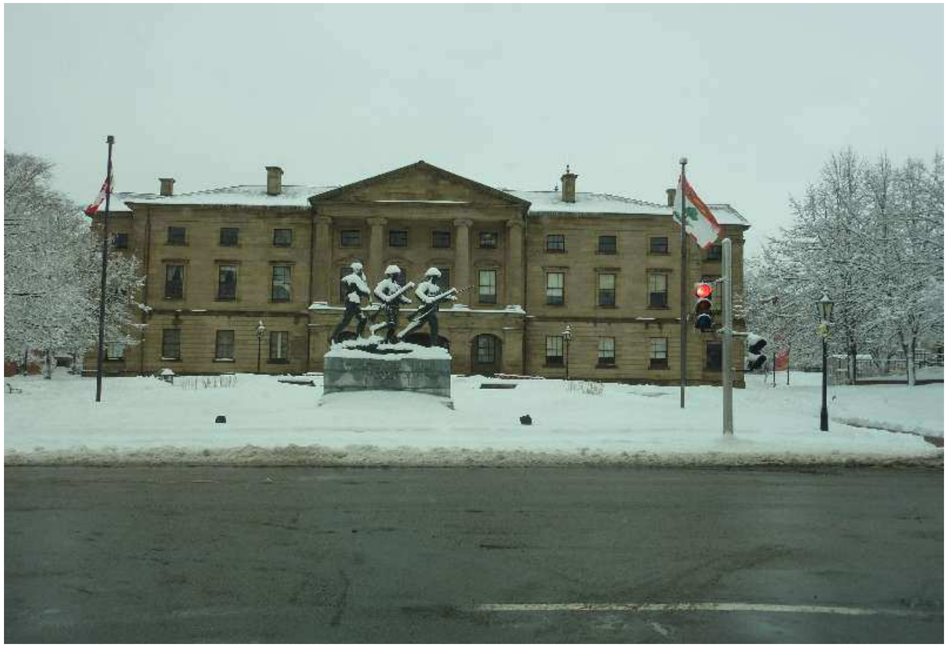
[5] Burgin, M.: Algorithmic complexity as a criterion of unsolvability, *Theoretical Computer Science* **383** (2007) 244–259

[6] Calude, C.: *Information and Randomness - An Algorithmic Perspective*, Springer-Verlag, Berlin (1994)

[7] Calude, C., Salomaa, K., Roblot, T.K.: Finite State Complexity and Randomness, *Technical Report CDMTCS* 374 December 2009/revised June 2010

La Mulţi Ani Cris!

Thank you for your attention!



← Charlottetown

and for the invitation