ON FORMAL DESCRIPTIONS OF CODE PROPERTIES

Stavros Konstantinidis (SK)
Saint Mary’s University, Halifax, CAN
(with Krystian Dudzinski)

presented at DCFS 2010 on Aug. 8, 2010
1 Introduction

Figure 1: Some developments in Coding Theory.
2 Codes as Formal Languages

- **Code.** A language $L$ that is uniquely decodable:

  $v_1 \cdots v_n = x_1 \cdots x_m \to m = n$ and $x_i = v_i$ for all $i$

  Decidable: [Sardinas, Patterson’53], [Levenshtein’61],
  [Berstel, Perrin’84], [Head, Weber’93], [McCloskey’96],
  [Fernau, Reinhardt, Staiger’07]

- **Code with decoding delay $d$.**

  $w \in vL^d\Sigma^* \cap x\Sigma^* \to x = v$.

  Decidable: [Levenshtein’61], [Devolder, Latt., Lit., Staiger’94],
  [SK’02]

- **Prefix code.** $v, vz \in L \to z = \lambda$ or $L \cap L\Sigma^+ = \emptyset$

- **Infix code.** $v, yvz \in L \to yz = \lambda$ or

  $L \cap (\Sigma^+ L\Sigma^* \cup \Sigma^* L\Sigma^*) = \emptyset$

  Decidable maximality: [Lam’98], [Kari, SK’05]
• **Thin language.** All words have different lengths

\[ x \in L, \ v \in L, \ |v| = |x| \rightarrow x = v. \]

• **Error-detecting language (wrt channel \( \gamma \)).**

\[ x \in L, \ (x, w) \in \gamma, \ w \in L \rightarrow x = w. \]

Decidable: [SK’02]

E.g. \((x, w) \in \gamma \) iff \( H(x, w) \leq 2. \)

— General decidability methods: [Head, Weber’93], [Jürgensen, Salomaa, Yu’94], [Jürgensen’99], [Domaratzki’04], [Kari,SK’05]

— General decidability of **maximality** methods: [Domaratzki’04], [Kari,SK’05], [Van’06]
3 Methodologies for Defining Code Properties

• Partial word orders. ‘≺_α’ strict, length increasing, transitive.

\[ L \in P_\alpha \text{ iff } \forall u, v \in L : \text{not} (u \prec_\alpha v) \]

[Shyr, Thierrin’77], [Van’06]

E.g. Prefix: \( u \prec_p v \text{ iff } v = uz \)

Theorem 1 \( P_{\text{code}} \) is not definable via any word order.

• n-ary word relations. [Jürgenesen, Yu’91]

‘ω_α ⊆ Σ × · · · × Σ’ upward symmetric.

Can use a ternary relation for comma-free codes:

\[ LL \cap \Sigma^+ L \Sigma^+ = \emptyset \]

Note that \( P_{\text{code}} \) is still not definable.
Dependence systems. [Jürgenesen, Yu’95]

Let \( n \in \mathbb{N} \cup \{\aleph_0\} \). We say that \( P_\alpha \) is an \( n \)-independence system if

\[
L \in P_\alpha \text{ iff } \forall L' \subseteq L \text{ with } |L'| < n : L' \in P_\alpha
\]

All previous properties are definable via independence systems. E.g., prefix codes are a 3-independence property and codes are \( \aleph_0 \)-independence.

**Theorem 2** Let \( P_\alpha \) be an independence property. Every \( P_\alpha \)-language is included in a maximal \( P_\alpha \)-language.

We use this methodology as a reference point.
4 Formal Methodologies

- To be compatible with and (when possible) more general than other existing methodologies.

- To be able to decide efficiently, given the description of a code property $\mathcal{P}$ and a regular language $L$, whether $L$ satisfies $\mathcal{P}$.

- To be able to decide (efficiently?), given the description of a code property $\mathcal{P}$ and a regular language $L$, whether $L$ is maximal with the property $\mathcal{P}$.

- To be able to build a LAnguage SERver that allows a user to enter descriptions of code properties and produce answers to questions about languages with the desired code properties.

**Theorem 3** Let $n \geq 2$. The class of $n$-independence properties is uncountable. In fact, already the class of $n$-independence properties whose elements are prefix codes is uncountable.

As any set of descriptions is countable, we cannot define/describe formally all possible independence properties.
• Some existing (semi-)formal methodologies.

{

Implicational conditions. [Jürgensen’99]

Example: suffix codes

\[ \varphi_s = \forall u, v, x : u \in L, v \in L, u = xv \rightarrow x = \lambda. \]

Trajectories. [Domaratzki’04],[Domaratzki,Salomaa’06]

Example: \( \hat{e} = 1^*0^* \) defines suffix codes

\[(L \Pi_e \Sigma^+) \cap L = \emptyset.\]

Language (in)equations. [Kari,SK’05]

Example: suffix codes

\[(L \leftrightarrow l_q \Sigma^+) \subseteq L^c \text{ iff } (L \leftrightarrow l_q \Sigma^+) \cap L = \emptyset.\]

In fact better described by a transducer...

}
5 Type-0 transducers

Rephrase definitions of prefix, suffix, infix code $L$:

\[ P(L) \cap L = \emptyset \quad S(L) \cap L = \emptyset \quad I(L) \cap L = \emptyset \]

Generalize: property defined by type-0 transducer $\hat{t}$

\[ \mathcal{P}_{0,i} = \{ L \subseteq M \mid \hat{t}(L) \cap L = \emptyset \}, \]

where

\[ \forall w \in M, \ w \notin \hat{t}(w) \]

and $M$ is our maximum set of words (e.g., $M = \Sigma^*$).

**Theorem 4** Every trajectory property is a type-0 transducer property (effectively), which in turn is a 3-independence property. There is a time $O(|\hat{t}| \cdot |\hat{a}|^2)$ decision algorithm, and maximality also is decidable.

**Corollary:** Decidable whether $L$ is maximal thin.
Figure 3: The 2-substitution-error-detection property.

6 Type-1 transducers

What about the properties of error-detection and -correction?

Property defined by type-1 transducer \( \hat{t} \)

\[
P_{1,t} = \{ L \subseteq M \mid \forall x \in L, \ \hat{t}(x) \cap (L - x) = \emptyset \},
\]

where

\[
\forall w \in M, \ w \in \hat{t}(w).
\]

**Theorem 5** Every type-0 transducer property is a type-1 property (effectively), which in turn is a 3-independence property. There is a polynomial time decision algorithm, and maximality also is decidable.

Based on deciding transducer functionality [Beal, Carton, Prieur, Sakarovitch’03]

Corollary: Maximality of error-detection and -correction decidable.
• **Error-correction.**

\[ x \in L, (x, w) \in \gamma, \text{ and } v \in L, (v, w) \in \gamma \rightarrow x = v. \]

A language is error-correcting for \( \gamma \) IFF it is error-detecting for \( \gamma^{-1} \circ \gamma \).
7 Decidability of Maximality

- Let $L$ be a language satisfying $\mathcal{P}_\alpha$. We have

\[
L \text{ is not maximal if and only if } \exists w \in M \setminus L : (L + w) \in \mathcal{P}_\alpha \\
\exists w \in M \cap L^c : w \notin R_\alpha(L) \quad \leftarrow (?) \\
M \cap L^c \cap R_\alpha(L)^c = \emptyset.
\]

- For both, $\mathcal{P}_{0,i}$ and $\mathcal{P}_{1,i}$, we have that

\[
R_\alpha(L) = \hat{i}(L) + \hat{i}^{-1}(L).
\]

When $L$ is given via an NFA, there is effectively an NFA for $L^c$ and $\hat{i}(L) + \hat{i}^{-1}(L)$.

- Unfortunately testing emptiness of

\[
L^c \cap (\hat{i}(L) + \hat{i}^{-1}(L))^c
\]

is PSPACE-complete, for given NFA and type-0 $\hat{i}$.

- Some questions.
  - What if we fix $\hat{i}$, e.g., the suffix code property?
  - What if $L$ is given via a DFA?
  - What’s the state complexity of $\hat{i}(L)$, $\hat{i}^{-1}(L)$, $\hat{i}(L) + \hat{i}^{-1}(L)$?
8 LA.SER. (in progress)

- **Current capability.** Web server accepting names of two files containing the automaton (language) and transducer (property) in Grail format, and returns whether the language satisfies the type-0 property.

  http://laser.cs.smu.ca/transducer/

- **Next capability.** Add translation from trajectory property to type-0 property.

- **Then.** Add type-1 transducer properties.

- **Then.** Extend to computing languages with desired properties. [Lam], [Van’06]
LA.SER. Architecture.

Figure 4: LA.SER. architecture.

http://www.djangoproject.com/
http://www.boost.org/
• Sample Grail file (automaton accepting $ab^*$):

(START) |- 1
1 a 2
2 b 2
2 -| (FINAL)
9 More theory in progress

What about decidability for properties like unique, or finite delay, decodability?

• **Look at this.** $L$ is a code IFF
\[
\forall x \in L, \ (\hat{t}_L^{-1} \circ \hat{t}_L)(x) \cap (L - x) = \emptyset,
\]
where $\hat{t}_L(w) = wL^*$.

Compare with $P_{1,i}$
\[
\forall x \in L, \ \hat{i}(x) \cap (L - x) = \emptyset.
\]

• A similar $\hat{t}_L$ works also for finite delay decodability. Unfortunately, $\hat{t}_L$ depends on $L$. 
NOTE: More references should be added...

References


[21] A.A. Sardinas and G.W. Patterson. A necessary and sufficient condition for the unique decomposi-


