Methods for Relativizing Properties of Codes

Helmut Jürgensen, Lila Kari, and Steffen Kopecki*

Abstract

The usual setting for information transmission systems assumes that all words over the source alphabet need to be encoded. The demands on encodings of messages with respect to decodability, error-detection, etc. are thus relative to the whole set of words. In reality, depending on the information source, far fewer messages are transmitted, all belonging to some specific language. Hence the original demands on encodings can be weakened, if only the words in that language are to be considered. This leads one to relativize the properties of encodings or codes to the language at hand.

We analyse methods of relativization in this sense. It seems there are four equally convincing notions of relativization. We compare those. Each of them has their own merits for specific code properties. We clarify the differences between the four approaches.

We also consider the decidability of relativized properties. If P is a property defining a class of codes and L is a language, one asks, for a given language C, whether C satisfies P relative to L. We show that in the realm of regular languages this question is mostly decidable.

1 Codes in Information Systems

In an information system, a source $\mathcal{S}$ generates messages[1] which, after some modifications, enter a channel $\mathcal{K}$. The channel may change a message because of physical errors or human interference or other reasons. For a given

*Department of Computer Science, The University of Western Ontario, London, Ontario, N6A 5B7

[1]On purpose we keep the notion of "message" and much of the other entities involved at an intuitive level. A formal treatment is found in [14]. Those details would be important for the detailed picture, but do not help with the main ideas.

channel $\mathcal{K}$, and an input message w, let $\kappa(w)$ be the corresponding set of potential output messages. Assume the output of the source is a message u and the corresponding input to the channel is a message $\gamma(u)$; then, as the output of the channel one may observe any message v in the set $\kappa(\gamma(u))$. The output of the channel undergoes changes again, resulting in $\delta(v)$, with the aim to recover the message originally sent as closely as possible. The technical details of this model are complicated [14]. Such details are provided in [14, 22]; instead, we explain the concepts and ideas intuitively only. We ask the reader not to make any assumptions beyond what is being stated as those might be quite misleading.

Coding theory in general assumes that a source can generate any sequence of output symbols, albeit with differing probabilities. In reality, a source may only generate a subset M of the set of all possible output sequences[2]. For instance, a source might generate exactly the grammatically correct sentences of a given natural language. For coding theory this changes important parts of the task. Instead of the set of all potential messages one only needs to deal with the messages in M: encode these messages and decode their channel outputs into messages in M.

Thus, suppose the source generates a message u in the set M. Technical modifications, which may include compression, encryption, encoding, and even modulation change the message u into the *sent message* $\gamma(u)$. This is what enters the channel. As output of the channel one finds a *received message* $v \in \kappa(\gamma(u))$ which may differ from $\gamma(u)$ due the physical characteristics of the channel $\mathcal{K}$. From v one tries to reconstruct a message $\delta(v) = u'$ such that $u' \in M$ and, ideally, such that $u' = u$. Given the characteristics of $\mathcal{S}$ and $\mathcal{K}$, the general goal is to find γ and δ such that the whole system works well, whatever this may mean concretely[3]. The choice of γ and δ implicitly depends on the set M.

In general we assume that all entities in the model use discrete signals and synchronized discrete time[4]. In particular this means that there are finite

[2]In a probabilistic setting, a threshold for the probability of a source output might determine the set M.

[3]For instance, if $\mathcal{S}$ and $\mathcal{K}$ are defined by probabilities, one may require the following: If $\mathcal{S}$ sends u and v is observed as the corresponding output, then the probability of u having been sent with v observed exceeds the probability of u' being sent when v is observed for all output messages of $\mathcal{S}$ different from u. For details of this probabilistic setting see [22]; for the corresponding combinatorial setting see [14].

[4]This latter assumption does not exclude synchronization errors on the logical level.

non-empty alphabets Θ and Σ such that the messages potentially issued by the source $\mathcal{S}$ form a language $M \subseteq \Theta^+$, where Θ^+ is the set of all (non-empty, finite) words which can be formed using the letters in Θ. Σ is the set of input symbols for $\mathcal{K}$ such that $\gamma(\Theta^+) \subseteq \Sigma^+$, where Σ^+ is the set of all non-empty words over Σ. Here γ need not be a mapping, but could be a relation $\gamma \subseteq \Theta^+ \times \Sigma^+$ with $\gamma(u) = \{u' \mid (u, u') \in \gamma\}$. Σ is also the set of output symbols of the channel[5]. κ is the input-output relation of the channel. Thus $(w, v) \in \kappa$ means that v is a potential output of κ for input w. The set $\kappa(w)$ for $w \in \Sigma^+$ may contain the empty word λ, hence

$$\kappa(w) = \{v \mid (w, v) \in \kappa\} \subseteq \Sigma^* = \Sigma^+ \cup \{\lambda\}.$$

In this setting δ is a partial mapping of Σ^* into Θ^+ such that, ideally, $\delta(\kappa(\gamma(u))) \in M$ for $u \in M$. In this context, we say that γ and δ are *encodings* and *decodings,* respectively. In general, $C = \gamma(\Theta) \subseteq \Sigma^+$ is called the code[6] of γ.

Ignoring many technical issues, γ encodes messages potentially sent by $\mathcal{S}$ and δ decodes received messages. The basic requirement is that $\delta(\gamma(u)) = u$ for all messages u. More subtle conditions may have to be satisfied, when errors need to be taken into account.

The successful functioning of such a system of information transmission depends very much on the properties of γ. In general we do not care about what happens to messages which will never be sent[7]. Hence, instead of considering the set Θ^* of all potential output messages over Θ, we focus on the set M of all potential (or likely) outputs of $\mathcal{S}$, but disregarding probabilities.

This simplifies the scenario: We eliminate the source $\mathcal{S}$ and the set of potential messages completely. Instead we consider a language $C \subseteq \Sigma^+$ serving as a code. The set M of potential messages is now replaced by the set $L \subseteq \Sigma^*$ of words which might have to be decoded as outputs of the channel.

[5]To use an output alphabet different from Σ certainly is an option, but is just a nuisance generalization, which changes little.

[6]Thus a code is just a subset of Σ^+ without any further requirements; in much, but not all of the literature, the term ‘code’ implies unique decodability. This issue is dealt with later in this paper.

[7]This is similar to a key argument in the proof of Shannon’s channel theorem (see [22], for example): Messages with probability 0 contribute errors of probability 0; hence we may ignore them and concentrate on the likely messages. Of course, messages with probability 0 can occur, but their influence has probability 0 too; hence, for practical purposes, they are ignored.

The precise relation between C and L will be discussed further below. Intuitively, the set $C^+ \cap L$ is the set of potential encoded messages, and L is the set potential channel outputs for these.

Finally we consider properties P of codes (or encodings) in this context. In general such a property would define the performance of an encoding in an information transmission setting such that the code itself determines properties of the encoding, for example: unique decodability; decoding delay; synchronization delay; error-detection; error-tolerance; error-correction. It turns out that such properties relativize in unexpected ways.

Obviously, when P contains a proposition of the form

$$\forall x_1, \ldots, x_n \in C \;\; \forall y_1, \ldots y_n \in \Sigma^+ \;\; \ldots,$$

replacing Σ^+ by the language L will change P. Intuitively, this is meant by relativizing properties of C to L.

With these preliminaries collected, we can state the main ideas of the present paper:

General Question. *Let X be a finite non-empty alphabet with at least two elements. Let L and C be non-empty languages over X. Let P be a property of languages.*

1. *Define what it means that C satisfies P relative to L.*
2. *With P fixed, what is the influence of L and vice versa?*
3. *Given P, C and L, can one decide whether C satisfies P with respect to L?*

To give this question a more concrete meaning, assume that P is the property of unique decodability: The set $C = \gamma(\Theta)$ is *uniquely decodable* if and only if every word in Σ^+ has at most one factorization into words in C; equivalently, C is uniquely decodable if and only if every word in C^+ has exactly one factorization into words in C. In general one implicitly assumes that $L = \Sigma^+$ or $L = C^+$ depending on the requirement 'at most one' versus 'exactly one'. To adapt the concept of unique decodability to the information system at hand, one would postulate only that $L \subseteq \Sigma^+$ and that each word in L have at most one factorization. In this case, C is uniquely decodable relative to L.

Example 1.1. *Let $\Sigma = \{a, b\}$, $L = (ab)^+$ and $C = \{a, ab, aab\}$. Then every word in L has a unique decoding with respect to C. On the other hand, the word aab has two distinct decodings. Hence C is not uniquely decodable in general, but uniquely decodable relative to L.*

Remark 1.1. *Let L and C be non-empty subsets of Σ^+. Every word in L has at most one factorization into words in C if and only if every word in $L \cap C^+$ has exactly one such factorization.*

Indeed, as every word in $L \cap C^+$ has a factorization into words in C and every word in L has no more than one such factorization, each word in $L \cap C^+$ has exactly one factorization. Conversely, as the words in $L \setminus C^+$ do not have a factorization at all, when the words in $L \cap C^+$ have unique factorizations, then each word in L has at most one factorization.

We now extrapolate from this idea to consider general code properties P as discussed in [14]. We only consider error-free communication via the channel $\mathcal{K}$. Thus $v = \gamma(u)$. The more general situation of errors will require several additional difficult steps of relativization, for which we do not have a sufficient answer yet.

Earlier work with the intent to relativize various properties of codes includes papers by Head [9, 10, 11, 12], Mahalingam [23], and by Daley, Jürgensen, Kari, and Mahalingam [2]. In the present paper we do not so much consider special cases, but focus on the relativization technique itself.

To define a class of codes two intuitively different techniques tend to be used: an essentially combinatorial approach, based mainly on the structure of words in the language C; an information theoretical approach, in which the coding and decoding functions are prevalent. For a example, a prefix code C over the alphabet Σ can be defined as a set of words, such that no word in the set is a proper prefix of another word in that set; this is the combinatorial view. Equivalently, C is a prefix code if it is uniquely decodable with decoding delay 0, the information theoretic view. Each of these definitions may lead to an intuitively convincing relativization. When these turn out not to be equivalent, which one should one choose? How are they related?

We focus on this fundamental issue: How to relativize code properties of either kind? When do revitalizations coincide? When is the relativized property decidable?

For classes of codes we refer primarily to [14]. Further information is found in [1] and [27, 31].

Our paper is structured as follows: In the next section we introduce the notation and basic notions. Most of this is standard, and included only to make the paper self-contained. Some of the main unrelativized concepts are explained in that part of the paper. In Section 3 we introduce and compare relativization methods. We review: (1) our approach of [2], which is based on a notion of admissibility; (2) the concepts proposed by Head [11]. This analysis leads to four essentially different, but equally well motivated, definitions of relativization. They are formally introduced in Section 3.3, where also their relationship, depending on the code property in question, is determined. Essentially, the four types of relativization arise from different views of how a code property might be violated when restricted to a set of messages smaller than Σ^+. While each of the four versions may be considered the "best" one, we only compare them, so as to understand what the respective strengths are. In Section 4 we consider decidability questions. Typically: Given C, L, P, and the type of relativization, we ask whether C is a code relative to L with property P and the given relativization method. The paper concludes with some general observations in Section 5.

There is a very important, but different, line of research which focuses on the relativization or generalization of just unique decodabilty. This traces back to work by Head and Weber [8, 30] and Harju and Karhumäki [7]. To our knowledge the most recent work in this field is a paper by Guzmán [6] and the thesis by Gümüştop [5].

2 Notation and Basic Notions

The sets of positive integers and of non-negative integers are $\mathbb{N}$ and $\mathbb{N}_0$, respectively. An alphabet is a non-empty set. To avoid trivial special cases, we assume that an alphabet has at least two elements. Throughout this paper Σ is an arbitrary, but fixed, alphabet. When required we add the assumption that Σ is finite. A word over Σ is a finite sequence of symbols from Σ; the set Σ^* of all words over Σ, including the empty word λ, is a free monoid generated by Σ with concatenation of words as multiplication. The set of non-empty words is Σ^+, that is, $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. A language over Σ is a subset of Σ^*. For a language $L \subseteq \Sigma^*$ and $n \in \mathbb{N}_0$ let

$$L^n = \begin{cases} \{\lambda\}, & \text{if } n = 0, \\ L, & \text{if } n = 1, \\ \{w \mid \exists u \in L \, \exists v \in L^{n-1} : w = uv\}, & \text{if } n > 1. \end{cases}$$

Moreover, let

$$L^* = \bigcup_{n \in \mathbb{N}_0} L^n \text{ and } L^+ = \bigcup_{n \in \mathbb{N}} L^n.$$

If P is a property of languages, then $\mathcal{L}_P(\Sigma)$ is the set of languages L over Σ for which $P(L) = 1$, that is, $P(L)$ is true. We write $\mathcal{L}_P$ instead of $\mathcal{L}_P(\Sigma)$ when Σ is understood. *In the remainder of this paper, unless explicitly stated otherwise, all languages are assumed to be non-empty.*

Many classes of codes and related languages can be defined systematically in terms of relations on the free monoid Σ^+ or in terms of abstract dependence systems. See [14, 16, 28, 31] for details. In the present paper only the following relations between words $u, v \in \Sigma^+$ are considered:

Property	*Definition*	Notation
u is a prefix of v:	$v \in u\Sigma^*$	$u \le_{\mathrm{p}} v$
u is a proper prefix of v:	$v \in u\Sigma^+$	$u <_{\mathrm{p}} v$
u is a suffix of v:	$v \in \Sigma^* u$	$u \le_{\mathrm{s}} v$
u is a proper suffix of v:	$v \in \Sigma^+ u$	$u <_{\mathrm{s}} v$
u is an infix of v:	$v \in \Sigma^* u \Sigma^*$	$u \le_{\mathrm{i}} v$
u is a proper infix of v:	$(u \le_{\mathrm{i}} v) \wedge (u \ne v)$	$u <_{\mathrm{i}} v$
u is an outfix of v:	$\exists u_1, u_2\, (u = u_1 u_2 \wedge v \in u_1 \Sigma^* u_2)$	$u\, \omega_{\mathrm{o}}\, v$
u is a proper outfix of v:	$(u\, \omega_{\mathrm{o}}\, v) \wedge (u \ne v)$	$u\, \omega_{\mathrm{o}}^{\ne}\, v$

We say that u is a *scattered subword* of v, and we write $u \le_{\mathrm{h}} v$, if, for some $n \in \mathbb{N}$, there are $u_1, u_2, \ldots, u_n \in \Sigma^*$ and $v_1, v_2, \ldots, v_{n+1} \in \Sigma^*$ such that $u = u_1 u_2 \cdots u_n$ and $v = v_1 u_1 v_2 u_2 \cdots u_n v_{n+1}$. We write $u <_{\mathrm{h}} v$ to denote the fact that u is a proper scattered subword of v, that is, $u \le_{\mathrm{h}} v$ and $u \ne v$. We say that u and v *overlap*, and we write $u\, \omega_{\mathrm{ol}}\, v$, if there is $q \in \Sigma^+$ such that $q <_{\mathrm{p}} u$ and $q <_{\mathrm{s}} v$ or vice versa. The relation ω_{ol} is symmetric. Note that a word can overlap itself.

To simplify or unify notation, we sometimes write ω_{p} instead of $\le_{\mathrm{p}}$ and so on, for the partial orders above.

A binary relation ω on Σ^+ defines the property (predicate) P_ω of languages[8] $L \subseteq \Sigma^+$ as follows: $P_\omega(L) = 1$ if and only if, for all $u, v \in L$, one has $u \not\omega\, v$ and $v \not\omega\, u$. Clearly, if $P_\omega(L) = 1$ and $L' \subseteq L$, then $P_\omega(L') = 1$. Thus $P_\omega(L) = 1$ if and only if $P_\omega(\{u, v\}) = 1$ for all $u, v \in L$. Here the

[8]The predicate P_ω asserts that L has a certain property, defined by the negation of a relation. Admittedly, this is awkward, but it is inevitable for reconciling the two different equally convincing approaches.

words u and v need not be distinct. This is important for the case of ω_{ol} for instance. Obviously, when ω is reflexive one has $P_\omega(L) = 0$ for every non-empty language L.

When $\omega = <_{\mathrm{p}}$ we write P_{p} instead of $P_{<_{\mathrm{p}}}$. Similarly, when $\omega = \omega_{\mathrm{ol}}$ we write P_{ol} instead of $P_{\omega_{\mathrm{ol}}}$. The predicates P_{s}, P_{i} and P_{o} are defined analogously starting from $<_{\mathrm{s}}$, $<_{\mathrm{i}}$ and $\omega_{\mathrm{o}}^{\neq}$, respectively.

For a set S, $\mathfrak{P}(S)$ is the set of all subsets of S and $\mathfrak{P}_{\mathrm{fin}}(S)$ is the set of all finite subsets of S. For $n \in \mathbb{N}$, let

$$\mathfrak{P}_{\leq n}(S) = \{T \mid T \in \mathfrak{P}(S), |T| \leq n\},\ \mathfrak{P}_{\geq n} = \{T \mid T \in \mathfrak{P}(S), |T| \geq n\}$$

and

$$\mathfrak{P}_{=n}(S) = \{T \mid T \in \mathfrak{P}(S), |T| = n\}.$$

In [14] the hierarchy of classes of codes is introduced using the systematic framework of abstract dependence systems. For the purposes of the present paper, the following simplified concepts suffice.

For the remainder of this section, we refer to [14, 31] and to sources cited there.

Let $C \subseteq \Sigma^+$. The language C is *uniquely decodable* if C^+ is a free subsemigroup of Σ^+ which is freely generated by C. A less abstract, but equivalent definition reads as follows:

Definition 2.1. *Let $C \subseteq \Sigma^+$ be a language over Σ, and let $w \in \Sigma^+$.*

1. *The word w is C-*decodable *if there are $n \in \mathbb{N}$ and words*

$$u_1, u_2, \ldots, u_n \in C \text{ such that } u_1 u_2 \cdots u_n = w.$$

   *In this case, the pair $(n, (u_1, u_2, \ldots, u_n))$ is called a C-*decoding *of w.*

2. *The language C is* uniquely decodable *if every word in Σ^+ has at most one C-decoding.*

Thus a language C is uniquely decodable, if and only if every word in C^+ has a unique C-decoding. We omit the reference to C when C is understood from the context. In the following we sometimes use parentheses to describe various C-decodings of a word. For example, if $C = \{a, ab, ba\}$, then $w = aba = (a)(ba) = (ab)(a)$ has two different C-decodings.

As every word in C^+ involves only finitely many elements of C, the language C is uniquely decodable if and only if every language in $\mathfrak{P}_{\mathrm{fin}}(C)$ is uniquely decodable.

In the literature one finds the term "code" used in two different ways: (1) *a non-empty language not containing the empty word;* (2) *a uniquely decodable non-empty language not containing the empty word.* For the rest of this paper we adopt the second meaning. By $\mathcal{L}_{\text{code}}$ we denote the set of codes over Σ. For a regular language $C \subseteq \Sigma^+$ it is decidable whether $C \in \mathcal{L}_{\text{code}}$; for linear languages the code property is undecidable.

We now introduce some important classes of languages or codes. Further classes will be defined when they are needed. Let $C \subseteq \Sigma^+$.

For $n \in \mathbb{N}$ with $n > 1$, C is an *n-code* if every language in $\mathfrak{P}_{\leq n}(C)$ is a code. In general, an n-code is not necessarily a code. By $\mathcal{L}_{n\text{-code}}$ we denote the set of n-codes over Σ. For regular C it is decidable whether $C \in \mathcal{L}_{2\text{-code}}$. For $\mathcal{L}_{3\text{-code}}$ the corresponding problem is open. The n-codes form an infinite descending hierarchy with $\mathcal{L}_{\text{code}}$ as its lower bound.

The language C is a *prefix code* if, for all $u, v \in C$, $u \not<_{\text{p}} v$. It is a *suffix code* if, for all $u, v \in C$, $u \not<_{\text{s}} v$. It is a *bifix code* if it is both a prefix code and a suffix code. It is an *infix code* if, for all $u, v \in C$, $u \not<_{\text{i}} v$. It is an *outfix code* if, for all distinct $u, v \in C$, $u \not\leq_{\text{o}} v$. It is a *solid code* if it is an infix code and if, for all $u, v \in C$ not necessarily distinct, u and v do not overlap. The language C is a *hypercode* if, for all distinct $u, v \in C$, $u \not<_{\text{h}} v$.

By $\mathcal{L}_{\text{p}}$, $\mathcal{L}_{\text{s}}$, $\mathcal{L}_{\text{b}}$, $\mathcal{L}_{\text{i}}$, $\mathcal{L}_{\text{o}}$, $\mathcal{L}_{\text{h}}$, and $\mathcal{L}_{\text{solid}}$ we denote the sets of prefix codes, suffix codes, bifix codes, infix codes, outfix codes, hypercodes, and solid codes, respectively. The first six of these classes of codes are defined by predicates P_{p}, P_{s}, P_{b}, P_{i}, P_{o} and P_{h} on $\mathfrak{P}_{=2}(C)$. For $\mathcal{L}_{\text{solid}}$ we need $P_{\text{solid}} = P_{\text{i}} \wedge P_{\text{ol}}$ on $\mathfrak{P}_{\leq 2}(C)$. We also use the predicates P_{code} on $\mathfrak{P}_{\text{fin}}(C)$ and $P_{n\text{-code}}$ on $\mathfrak{P}_{\leq n}(C)$ defining $\mathcal{L}_{\text{code}}$ and $\mathcal{L}_{n\text{-code}}$, respectively.

For $n \in \mathbb{N}$, the language C is an *intercode of index n* if, $\Sigma^+ C^n \Sigma^+ \cap C^{n+1} = \emptyset$. The class $\mathcal{L}_{\text{inter}_n}$ of intercodes of index n is defined by a predicate P_{inter_n} on $\mathfrak{P}_{\leq 2n+1}(C)$ derivable from P_{i}. The set $\mathcal{L}_{\text{inter}_1}$ of intercodes of index 1 is exactly the set $\mathcal{L}_{\text{comma-free}}$ of *comma-free codes.* The languages in $\mathcal{L}_{\text{inter}} = \bigcup_{n \in \mathbb{N}} \mathcal{L}_{\text{inter}_n}$ are called *intercodes.*

Lemma 2.1. (See [14, 31]) *The following inclusions hold:*

$$\mathcal{L}_{\text{p}} \cup \mathcal{L}_{\text{s}} \subsetneq \mathcal{L}_{\text{code}},\ \mathcal{L}_{\text{i}} \cup \mathcal{L}_{\text{o}} \subsetneq \mathcal{L}_{\text{b}} = \mathcal{L}_{\text{p}} \cap \mathcal{L}_{\text{s}},$$

$$\forall n\ \mathcal{L}_{\text{inter}_n} \subsetneq \mathcal{L}_{\text{inter}_{n+1}} \subsetneq \mathcal{L}_{\text{inter}} \subsetneq \mathcal{L}_{\text{b}},\ \mathcal{L}_{\text{h}} \cap \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{h}} \subsetneq \mathcal{L}_{\text{i}} \cap \mathcal{L}_{\text{o}}$$

and

$$\mathcal{L}_{\text{h}} \cap \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{comma-free}} \subsetneq \mathcal{L}_{\text{i}}.$$

It will simplify the notation significantly and also open the prospects of considering a different set of problems if we weaken the definitions as follows: For

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, n\text{-code}, \text{comma-free}\}$$

and potentially other types ϱ of language properties, P_ϱ is a predicate on $\mathfrak{P}_{\mathrm{fin}}(C)$ in the following sense: A language $L \subseteq C$ has the property ϱ if and only if $P_\varrho(L)$ holds true, that is, $P_\varrho(L) = 1$; for $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{solid}, \mathrm{ol}\}$ we are mainly interested in situations when $|L| \leq 2$ as this leads to manageable decision properties. As a warning to the reader – we have seen this misread before – the set $\{u, v\}$ is equal to $\{u\}$ when $u = v$, that is, $\{u, v\}$ is not a pair, but a set.

3 Variations of Definitions

The definition of relativized codes given in [2] was phrased so as to capture and generalize the special definitions proposed by Head in [9, 10, 11, 12] in the more general framework of relations or predicates described in [14]. As noted in [2] these definitions differ in a subtle way.

In Sections 3.1 and 3.2, we review two natural proposals for relativizing code concepts. Abstracting from these, and considering other likely scenarios, it turns out that one has to consider at least four versions according to the phenomena by which violations of code properties could manifest themselves, each of them well motivated. These are investigated in detail in Section 3.3 as violation-freeness or admissibility of words. In Section 3.4 relativized codes are defined and inclusions between classes of relativized are proved. We compare the concepts considered in the earlier work [2, 11] to the ones introduced in the present paper in Section 3.5.

3.1 Admissibility of Words as Defined in [2]

We review the definitions and discussions of [2]. An improved general framework is proposed in Section 3.3.

Definition 3.1. *Let C be a subset of Σ^+ and let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. A word $q \in C^+$ is said to be P-*admissible *for C if the following condition is satisfied: if $q = xuy = x'u'y'$, with $u, u' \in C$ and $x, x', y, y' \in C^*$ then $P(\{u, u'\}) = 1$.*

This means that a word $q \in C^+$ is P-admissible if every two words $u, u' \in C$ appearing in C-decodings of q, together satisfy the property P. For example, for $P = P_{\mathrm{p}}$, a word q is *prefix-admissible,* if no two words $u, u' \in C$ appearing in C-decodings of q are strict prefixes of each other. There is a subtle point: Suppose that u' is a proper prefix of u. For a word q, three different situations need to be considered:

1. The word q has a C-decoding of the form
$$\cdots(u')\cdots(u)\cdots \text{ or } \cdots(u)\cdots(u')\cdots.$$
2. The word q has two C-decodings of the forms
$$\cdots(u')\cdots \text{ and } \cdots(u)\cdots.$$
3. The word q has two C-decodings of the form $q_1(u')v'q_2$ and $q_1(u)q_2$ with $u = u'v'$ where $q_1, q_2, v'q_2 \in C^*$, $v' \in \Sigma^+$.

The difference between these situations becomes apparent in our discussion of relativized solid codes below. Definition 3.1 applies to any occurrences of u and u', not just to those situations in which u and u' start at the same position in q, and also not just to occurrences of u and u' in the same C-decoding of q. Thus, if u and u' are distinct and occur in any C-decodings of a word $q \in L$, which is prefix-admissible for C, then the set $\{u, u'\}$ must be a prefix code.

Similarly, a word $q \in C^+$ is *overlap-admissible* if no two words $u, u' \in C$, not necessarily distinct and appearing in any C-decodings of q, overlap. In particular, if $u \in C$ and u occurs in a C-decoding of q, then u must not overlap itself.

Definition 3.2. *Let C be a subset of Σ^+, let $L \subseteq C^+$ and let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. Then C is said to* satisfy P relative to L *if every $q \in L$ is P-admissible for C.*

Definition 3.3. *When C satisfies P relative to L we say that C is a P*-code relative to *L.*

As the predicate P is arbitrary, a P-code relative to L need not be uniquely decodable even when $L = C^+$. The restriction of L being a subset of C^+ turns out to be too restrictive in the new context of this paper and is lifted starting in Section 3.3.

The following trivial observation is used without special mention in the sequel.

Remark 3.1. *Let P, P_1 and P_2 be predicates on $\mathfrak{P}_{\leq 2}(C)$ with $P = P_1 \wedge P_2$. Let q, C and L be as in Definitions 3.1, 3.3 and 3.2. The following statements hold true:*

1. *q is P-admissible for C if and only if q is both P_1-admissible and P_2-admissible for C.*

2. *C satisfies P relative to L if and only if C satisfies P_1 and P_2 relative to L.*

3. *C is a P-code relative to L if and only if C is both a P_1-code and a P_2-code relative to L.*

3.2 Definitions Inspired by Tom Head

In [11] and related papers, Head proposed various relativizations of code concepts. The most relevant for the present discussion, because it introduces issues not encountered in other contexts, is that of relativized solid codes. The formalism used here leads to a novel general concept of relativization. This section of the paper summarizes ideas and statements from [2] relevant to the issue at hand.

Definition 3.4. ([9]) *Let C and L be non-empty subsets of Σ^+. The set C is a* solid code relative to L *if it satisfies the following conditions for all words $q \in L$:*

1. *if $q = xszty$ with $x, y, s, t \in \Sigma^*$ such that $z, szt \in C$, then $st = \lambda$;*

2. *if $q = xszty$ with $x, y, s, t \in \Sigma^*$ such that $sz, zt \in C$ and $z \in \Sigma^+$ then $st = \lambda$.*

The first condition states that, for $u, v \in C$, if $u <_{\mathrm{i}} v$, then, for all $q \in L$, $v \not\leq_{\mathrm{i}} q$. The second condition states that if $u, v \in C$, and u and v overlap as $u = sz$ and $v = zt$ with $z \in \Sigma^+$, then, for all $q \in L$, $szt \not\leq_{\mathrm{i}} q$.

Definition 3.4 is one possible relativization of the notion of solid code. It differs from the notion of P_{solid}-code relative to a language as introduced in Definition 3.3.

Note that, if C is a solid code relative to L then C is a P_{i}-code relative to $L \cap C^+$. Indeed, let q in $L \cap C^+$. If $u \in C$ occurs in a C-decoding of q, $v \in C$ and $u <_{\mathrm{i}} v$, then $v \not<_{\mathrm{i}} q$. Hence v does not occur in a C-decoding of q. As shown in Example 3.1 below, C being a solid code relative to L does not imply that C is a P_{ol}-code or a P_{solid}-code relative to L.

For (unrelativized) solid codes there is also a definition based on decompositions of messages (see [14]): Let C be a subset of Σ^+ and $q \in \Sigma^+$. A *C-decomposition of q* consists of two sequences $u_0, u_1, \ldots, u_n \in \Sigma^*$ and $v_1, v_2, \ldots, v_n \in C$ for some $n \in \mathbb{N}$, such that $q = u_0 v_1 u_1 v_2 u_2 \cdots v_n u_n$ and $v \not\leq_{\mathrm{i}} u_i$ for all $v \in C$ and $i = 0, 1, \ldots, n$. Every word $q \in \Sigma^+$ has at least one C-decomposition. Note that every C-decomposition of a word in C^+ can be considered as a C-decoding as follows:

$$u_0 = u_1 = \cdots = u_n = \lambda$$

and the C-decoding is

$$(n, (v_1, v_2, \ldots, v_n)).$$

The set C is a solid code if and only if every word in Σ^+ has a unique C-decomposition. In [13], a relativization of the notion of solid code is proposed, which is based on the uniqueness of C-decompositions, and this notion turns out to be equivalent to the one of Definition 3.4.

Proposition 3.1. ([13]) *Let $L \subseteq \Sigma^+$. A language $C \subseteq \Sigma^+$ is a solid code relative to L if and only if every word $q \in L$ has a unique C-decomposition.*

The difference between these equivalent concepts and our approach to relativizing solid codes is illustrated by the following example.

Example 3.1. ([11]) *Let $\Sigma = \{a, b, c\}$ and $C = \{ab, c, ba\}$. The set C is not overlap-free, hence not a solid code. By Definition 3.4, C is a solid code relative to the language $L = (\{abc\} \bigcup \{cba\})^*$. However, the set C is not a P_{solid}-code relative to L, as $q = abccba \in L$ has the C-decoding $(ab)(c)(c)(ba)$ and is thus not P_{solid}-admissible since $ab\, \omega_{\mathrm{ol}}\, ba$.*

The main differences between Definitions 3.3 and 3.4 are as follows:

1. According to Definition 3.3, the mere and possibly unrelated existence of words for which the predicate is false constitutes a violation. According to Definition 3.4, the words in question must be in a specific violating position.

2. According to Definition 3.3, the words in violation must occur in C-decodings. According to Definition 3.4, they may appear anywhere.

In the next section, we analyse these differences and provide new definitions according to the analysis. Altogether, we have to investigate four different ways in which code concepts can be relativized.

3.3 Violating Instances

There does not seem to be a unique best scheme for relativizing code properties. All proposed schemes seem to diverge not only when the language L relativized to is a subset of Σ^+ or of C^+, but also when the particular types of violations of the code properties are considered. We now identify four violating scenarios in very general terms. These seem to be the most common ones in real systems. For specific natural code properties we state their relativizations. We also determine the connection between the four notions of violation. Our basic definitions may seem to be far too general; this permits us to capture most of the interesting cases and to leave the field open for other cases which might require a relativization as well.

To clarify the intuition, we start with examples. We consider a language $C \subseteq \Sigma^+$ and a predicate P defining a class of codes.

A violating instance of P_{p}, the prefix-freeness predicate, would be the occurrence of a word $v \in C$ such that there is a word $u \in C$ with $u <_{\mathrm{p}} v$, that is, $P_{\mathrm{p}}(\{u, v\}) = 0$. For P_{s}, P_{i}, P_{h}, P_{o} and several other such predicates we have analogous characterizations. To help the readers' intuition we switch freely between predicates and relations whenever one or the other seems easier to understand.

Take P_{b}. One has $P_{\mathrm{b}}(\{u, v\}) = 0$ if $u <_{\mathrm{p}} v$ or $u <_{\mathrm{s}} v$ or vice versa. Thus there are two potential violating instances of P_{b}, manifested as violating instances of P_{p} and P_{s}, respectively.

This seems to determine the pattern for predicates defined by conjunctions or disjunctions of predicates.

Thus a violating instance of the conjunction (intersection) P of two predicates P_1 and P_2 could be a violating instance of P_1 or a violating instance of P_2. Dually, if P is defined as the disjunction (union) of two such predicates, then violating instances of P are exactly the instances which are violating both P_1 and P_2. This idea works well also with $P_{\mathrm{solid}} = P_{\mathrm{i}} \wedge P_{\mathrm{ol}}$.

These considerations suggest the following tentative definition:

Let P be an n-ary predicate. A violating instance *of P is a set $\{u_1, \ldots, u_n\}$ of words such that $P(\{u_1, \ldots, u_n\}) = 0$.*

This definition is not good enough as it does not capture how the words in question are actually located with respect to each other; hence, a proper definition needs to be based on relations or tuples with special properties rather than sets.

We consider a set of more detailed examples in order to detect a pattern. For

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}$$

and potentially other types ϱ of language properties, let ω_ϱ or $<_\varrho$ be the corresponding relation or partial order, and P_ϱ be the corresponding predicate. Let $C \subseteq \Sigma^+$.

1. A violating instance of P_{p} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u <_{\mathrm{p}} v$. Similarly for P_{s}, P_{pi}, P_{si}, and P_{i}.

2. A violating instance of P_{b} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u \, \omega_{\mathrm{b}} \, v$, that is, $u <_{\mathrm{p}} v$ or $u <_{\mathrm{s}} v$.

3. A violating instance of P_{o} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u \, \omega_{\mathrm{o}} \, v$ and $u \neq v$.

4. A violating instance of P_{ol} is the occurrence of a word $w = w_1w_2w_3$ with $w_1, w_2, w_3 \in \Sigma^+$ such that $w_1w_2 \in C$ and $w_2w_3 \in C$; thus, $w_1w_2 \, \omega_{\mathrm{ol}} \, w_2w_3$.

5. A violating instance of P_{solid} is the occurrence of a word which is a violating instance of P_{i} or of P_{ol}.

6. A violating instance of P_{inter_n} is the occurrence of a word

   $$w = v_1v_2 \cdots v_{n+1} \text{ with } v_1, v_2, \ldots, v_{n+1} \in C$$

   such that there are words

   $$u_1, u_2, \ldots, u_n \in C \text{ and } x, y \in \Sigma^+$$

   with $xu_1u_2 \cdots u_ny = w$.

7. A violating instance of $P_{\text{comma-free}}$ is the occurrence of a word $w = v_1 v_2$ such that there are words $u \in C$ and $x, y \in \Sigma^+$ with $xuy = w$.

8. A violating instance of P_{h} is the occurrence of a word $v \in C$ such that there is a word $u \in C$ with $u <_{\text{h}} v$.

The cases (1), (2), (3), (6), (7), and (8) above have in common that the (proper) relation involved has a "direction": The relations for

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}\}$$

are anti-symmetric. For $\varrho \in \{\text{inter}_n, \text{comma-free}\}$, that is, cases (6) and (7), one considers the relations ω_{inter_n} and $\omega_{\text{comma-free}}$ defined as follows[9] [14]:

- ω_{inter_n} is a $(2n+1)$-ary relation on C such that

  $$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\text{inter}_n}$$

  if and only if there are $x, y \in \Sigma^+$ such that $v_1 v_2 \cdots v_{n+1} = x u_1 u_2 \cdots u_n y$.

- $\omega_{\text{comma-free}} = \omega_{\text{inter}_n}$ for $n = 1$.

We interpret ω_{inter_n} as a binary relation between n-tuples and $(n+1)$-tuples of words in C. Let $\overline{\omega_{\text{inter}_n}}$ be this binary relation, that is,

$$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\text{inter}_n}$$

if and only if

$$((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_{n+1})) \in \overline{\omega_{\text{inter}_n}}.$$

Similarly, we obtain $\overline{\omega_{\text{comma-free}}}$ from $\omega_{\text{comma-free}}$. Then, by definition, both $\overline{\omega_{\text{inter}_n}}$ and $\overline{\omega_{\text{comma-free}}}$ are anti-symmetric binary relations.

For P_{ol}, instead of considering a binary relation between code words, it seems more adequate to consider a binary relation $\overline{\omega_{\text{ol}}}$ between a pair (u_1, u_2) of codewords and a word $w \in \Sigma^+$ such that $(u_1, u_2)\ \overline{\omega_{\text{ol}}}\ w$ if and only if there are $w_1, w_2, w_3 \in \Sigma^+$ such that $u_1 = w_1 w_2$, $u_2 = w_2 w_3$, and $w_1 w_2 w_3 = w$.

One could apply similar modifications to the relations defining the outfix codes, the hypercodes, and all the codes in the shuffle hierarchy. For

[9]In [14] the order of the components is different. The change is not essential, but simplifies the presentation.

example, instead of $<_{\mathrm{h}}$ one could use the relation $\overline{\omega_{\mathrm{h}}}$ defined as follows: $(u_1, u_2, \ldots, u_k)\ \overline{\omega_{\mathrm{h}}}\ (v)$ if and only if

$$v \in \Sigma^* u_1 \Sigma^* u_2 \cdots \Sigma^* u_k \Sigma^* \text{ and } u_1 u_2 \cdots u_k \neq v,$$

where $k \in \mathbb{N}$ and $u_1, u_2, \ldots, u_k, v \in \Sigma^+$. For the present purposes the following, less intuitive, alternative

$$(u_1, u_2, \ldots, u_k)\ \overline{\omega_{\mathrm{h}}}\ (v) \text{ if and only if } u_1 u_2 \cdots u_k <_{\mathrm{h}} v$$

would also work. The former captures the idea that $u_1 \leq_{\mathrm{i}} q$, $u_2 \leq_{\mathrm{i}} q$, $\ldots$, $u_k \leq_{\mathrm{i}} q$ in the order given by the k-tuple $(u_1, u_2, \ldots, u_k)$. The latter is a simple reformulation of the embedding order. For our purposes, neither modification is needed.

Note that the transition from a relation ω_ϱ to its overlined version $\overline{\omega_\varrho}$ is *ad hoc* and not claimed to be in any way defined by an operator. We introduce the latter only for convenience. In the sequel, to keep the notation simple, we drop the distinction when there is no risk of confusion. For example, a statement of the form

"For $\varrho \in \{\mathrm{p}, \ldots, \mathrm{inter}_n, \ldots\}$ the relation ω_ϱ satisfies ..."

refers to ω_{p} for $\varrho = \mathrm{p}$ and, depending on the context, to $\omega_{\mathrm{inter}_n}$ or to $\overline{\omega_{\mathrm{inter}_n}}$ for $\varrho = \mathrm{inter}_n$.

To define violating instances in rather general terms, we consider binary relations on tuples of words and their corresponding binary predicates.

For any set S and any $n \in \mathbb{N}$, let $n\text{-}\mathsf{tuples}(S)$ be the set of n-tuples of elements in S and let $\mathsf{all\text{-}tuples}(S) = \bigcup_{n \in \mathbb{N}} n\text{-}\mathsf{tuples}(S)$. We consider binary relations ω between tuples of words over Σ. Typically there is a small upper bound on the arity of the tuples. Such a relation ω would be a subset of

$$\bigcup_{1 \leq k \leq m} \bigcup_{1 \leq n \leq m} k\text{-}\mathsf{tuples}(\Sigma^*) \times n\text{-}\mathsf{tuples}(\Sigma^*)$$

for some $m \in \mathbb{N}$. In some quite natural situations however, like that of hypercodes, there might not be *a priori* bounds on k and n. This concern will be kept in mind as we propose definitions. As such relations are defined by (disjoint) unions of relations in a natural way as expressed by the formula above, their respective properties are conjunctions of the individual properties according to the constituents. The details are explained by example below.

Definition 3.5. *Let $n \in \mathbb{N}$ and let $\mathfrak{u} = (u_1, u_2, \ldots, u_n)$ be an n-tuple of words in Σ^*. Define $\mathsf{word}(\mathfrak{u})$ as the word $u_1u_2 \cdots u_n$. Moreover, for $u \in \Sigma^*$, let $\mathsf{word}(u) = u$.*

The present goal is as follows: Let $C \subseteq \Sigma^+$, $C \neq \emptyset$. For a word $q \in \Sigma^+$ we want to express that q does not contain words in C which violate a binary relation ω on $\mathsf{all\text{-}tuples}(\Sigma^+)$, the latter defining a class of languages or codes. Additionally, if $\mathfrak{u}\ \omega\ \mathfrak{v}$, then $\mathsf{word}(\mathfrak{u})$ and $\mathsf{word}(\mathfrak{v})$ must appear in some "natural" relationship within q. A first attempt towards this goal might read as follows:

> Let ω be a binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$ and let $q \in \Sigma^+$. A *violating instance of ω in q* is a pair $(\mathfrak{u}, \mathfrak{v})$ of distinct tuples of words in Σ^+ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$ and $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$.

At a first glance this seems to be a clean definition. It only involves the relation ω, but not the set C, and the latter can be built in later as a constraint. The following example shows that the attempted definition will not work without a connection to C.

Example 3.2. *Let $\Sigma = \{a, b\}$ and $\omega = <_{\mathrm{p}}$. Then every word of length at least 2 contains a violating instance of $<_{\mathrm{p}}$.*

Nevertheless, we work with this intuition. It does not lead to a general definition, but at least to a usable one for many relevant cases. To simplify terminology, when $(\mathfrak{u}, \mathfrak{v})$ is a violating instance of ω in q in the tentative sense above, we also say, equivalently, that *q contains $(\mathfrak{u}, \mathfrak{v})$ as a violating instance of ω* – or of the predicate P_ω defining ω.

For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$ we just consider the relation ω_ϱ. Similarly, for the relations defining the shuffle hierarchy. For $\varrho \in \{\mathrm{inter}_n, \text{comma-free}, \mathrm{ol}\}$, the relations $\overline{\omega_{\mathrm{inter}_n}}$, $\overline{\omega_{\text{comma-free}}}$, and $\overline{\omega_{\mathrm{ol}}}$ will serve. Thus, also the solid codes are included. In each of the cases considered here, $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$ implies that each component of $\mathfrak{u}$ is a subword, possibly scattered, of q. Our present motivation was to cover as much as possible of the code hierarchy of [14].

To address the problems with the notion of violating instance of a relation ω, we consider, simultaneously, a relation ω, a non-empty set C of words in Σ^+, and a word $q \in \Sigma^+$. The relation ω is meant to describe a class of languages – or codes – such that C does not contain any words which would lead to a violating instance in q. Without loss of generality, one can assume that ω is irreflexive. We did not find a satisfactorily simple definition which could be applied to any binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$. Especially relations like ω_{ol} or $\overline{\omega_{\mathrm{ol}}}$ cause difficulties, as the relative positions of the occurrences of their components in a word are not fixed. Therefore, from here on we consider only a restricted class of relations:

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}.$$

Definition 3.6. *Let $C \subseteq \Sigma^+$ and $q \in \Sigma^+$. Let $\omega \neq \omega_{\mathrm{ol}}$ be an irreflexive, binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$ such that, for all $\mathfrak{u}, \mathfrak{v} \in \mathsf{all\text{-}tuples}(\Sigma^+)$, $\mathfrak{u}\ \omega\ \mathfrak{v}$ implies $\mathsf{word}(\mathfrak{u}) \leq_{\mathrm{h}} \mathsf{word}(\mathfrak{v})$. Let P_ω be the predicate defining ω.*

1. *The word q is P_ω-*violation-free for decompositions with respect to C, *if there are no $\mathfrak{u}, \mathfrak{v} \in \mathsf{all\text{-}tuples}(C)$ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$ and $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$.*

2. *The word q is P_ω-*violation-free for decodings with respect to C, *if for all $q_1, q_2 \in C^*$ and all $\mathfrak{v} \in \mathsf{all\text{-}tuples}(C)$ with $q = q_1\mathsf{word}(\mathfrak{v})q_2$ there is no $\mathfrak{u} \in \mathsf{all\text{-}tuples}(C)$ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$.*

3. *The word q is said to be P_{ol}-*violation-free for decompositions with respect to C, *if there are no words $u, v, w \in \Sigma^+$ such that $uv, vw \in C$ and $uvw \leq_{\mathrm{i}} q$.*

4. *The word q is said to be P_{ol}-*violation-free for decodings with respect to C, *if there are no words $q_1, q_2 \in C^*$ and $u, v, w \in \Sigma^+$ with $uv, vw \in C$ such that $q = q_1uvq_2$ with $w \leq_{\mathrm{p}} q_2$ or $q = q_1vwq_2$ with $u \leq_{\mathrm{s}} q_1$.*

To explain Definition 3.6, we consider the special cases of prefix codes, outfix codes, intercodes of some index n, and solid codes defined by the relations $<_{\mathrm{p}}$, $\omega_{\mathrm{o}}^{\neq}$, $\omega_{\mathrm{inter}_n}^{\neq}$, and ω_{solid} as characteristic examples. Most other cases in the hierarchy of codes are analogous. In the definition we attempt to capture an essential idea of Head's relativization: the respective code property is violated if and only if the words involved appear exactly in the relative positions as defined by the code property. Beyond that, we distinguish between violating instances for decompositions and violating instances for decodings. The former may occur anywhere in the word q under consideration – and this is the case of Head's definition (Definition 3.4); the latter can only occur at positions defined by a decoding. This distinction turns out to be important, as fewer positions in a word under consideration need to be examined in the case of decodings compared to the case of decompositions.

Head's relativization of the condition of overlap-freeness in Definition 3.4 really applies only to decompositions. In Definition 3.6(4) we propose a possible interpretation of Head's approach in the context of decompositions. Another possible interpretation would be as follows.

> Let $q = q_1uvwq_2 \in C^*$ with $uv, vw \in C$ and $u, v, w \in \Sigma^+$. Then $q_1, wq_2 \in C^*$ or $q_2, q_1u \in C^*$.

This is equivalent to Definition 3.6(4).

The first two parts of Definition 3.6 refer to binary relations ω on tuples of words in Σ^+ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$ implies that $\mathsf{word}(\mathfrak{u}) \leq_{\mathrm{h}} \mathsf{word}(\mathfrak{v})$. Thus, if $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$, then $\mathsf{word}(\mathfrak{u})$ appears as a possibly scattered subword of that occurrence of $\mathsf{word}(\mathfrak{v})$ in q. The relation ω_{ol} is an important example of a relation which does not fit into this pattern. We include ω_{ol} as a special case in Definition 3.6 to exhibit unsolved problems in the relativization methods and the need for a more inclusive approach.

Among the cases for illustrating Definition 3.6, a simple one is that of prefix codes and the like. The class of codes C is defined by a partial order $\omega^{\neq}$ on Σ^+ such that $u, v \in C$ implies that $u \not\omega^{\neq} v$. Moreover, $u\ \omega^{\neq}\ v$ implies $u <_{\mathrm{i}} v$. If q contains a violation of ω, then there are $u, v \in C$ such that $u\ \omega^{\neq}\ v$ and $v \leq_{\mathrm{i}} q$. Thus the mere occurrence of v as an infix of q results in a violating instance, for decompositions. For decodings, the word v has to appear at a special spot, determined by a decoding; but note that the decoding need not be unique.

The case of outfix codes and of all shuffle codes down to the class of hypercodes requires special consideration as to what we mean by "violation". The case of outfix codes is indicative of the issues. Suppose u is a proper outfix of v. Then $v = u_1v_0u_2$ with $u_1u_2 \in \Sigma^+$, $u = u_1u_2$, and $v_0 \in \Sigma^+$. If $v \leq_{\mathrm{i}} q$ we have a violating instance according to the definition, but $u \not\leq_{\mathrm{i}} q$. Do we want this? We argue as follows: The intent of using an outfix code could be to detect insertion errors, like the ones which change u into v. In this, clearly, the occurrence of v in q gives rise to an ambiguity as to how q should be read (both for decompositions and decodings). Similar arguments concern the whole shuffle hierarchy and motivate the condition of $\mathsf{word}(\mathfrak{u}) \leq_{\mathrm{h}} \mathsf{word}(\mathfrak{v})$ above. In general, the embedding is completely determined by ω.

The case of intercodes of index n is special only in that we deal with tuples of words. The relation defining the intercodes satisfies the conditions trivially.

Finally, for solid codes we need to consider the relation $\omega_{\mathrm{solid}} = <_{\mathrm{i}} \cup \omega_{\mathrm{ol}}$. The rôle of $<_{\mathrm{i}}$ is similar to that of the prefix order above. The rôle of ω_{ol} is different. Regardless of whether we use ω_{ol} or $\overline{\omega_{\mathrm{ol}}}$, there is a problem which seems to require special measures.

- Using ω_{ol}: If $u \ \omega_{\mathrm{ol}} \ v$ with $u, v \in \Sigma^+$ then $v \leq_{\mathrm{i}} q$ does not imply that $u \leq_{\mathrm{h}} q$.
- Using $\overline{\omega_{\mathrm{ol}}}$: If $(u_1, u_2) \ \overline{\omega_{\mathrm{ol}}} \ v$ then $v \leq_{\mathrm{i}} q$ does not imply $\mathsf{word}((u_1, u_2)) = u_1 u_2 \leq_{\mathrm{h}} q$. However, we have $u_1 \leq_{\mathrm{i}} q$ and $u_2 \leq_{\mathrm{i}} q$.

In either case, the mere occurrence of v does not result in a violating instance in general.

Example 3.3. *Let $\Sigma = \{a, b\}$.*

1. *Consider the prefix order $<_{\mathrm{p}}$ and the language $C = \{a, ab\}$. This language is not a prefix code. The set of words which are violation-free of $<_{\mathrm{p}}$ for decompositions with respect to C are the words not containing ab, that is, all the words in $\Sigma^+ \setminus \Sigma^* ab \Sigma^* = a^+ \cup b^+ a^*$. The set of words which are violation-free of $<_{\mathrm{p}}$ for decodings with respect to C are the words in $\Sigma^+ \setminus C^* ab C^*$.*
2. *For the outfix relation, consider the language $C = \{aa, aba\}$ which is not an outfix code. A violation-free word for decompositions must not contain aba as an infix, that is, must be in $\Sigma^+ \setminus \Sigma^* aba \Sigma^*$. For decodings, such a word must not have the form $C^* aba C^*$.*
3. *For the intercode relation of index n, consider, without loss of generality, $n = 1$ and the language $C = \{ab, bba\}$. The language C is not an intercode of index 1, that is, not a comma-free code, as $\Sigma^+ C \Sigma^+ \cap C^2 \neq \emptyset$ with ab and $bbabba$ as witnesses. Note that C is a bifix code. For decompositions, $bbabba$ must not occur as an infix. For decodings, any word not in $C^* bbabba C^*$ is violation-free.*
4. *For the solid code relation, the infix part is analogous to $<_{\mathrm{p}}$ that has already been illustrated. The "new" problem is that of overlaps. Consider $C = \{ab, ba\}$, which is an infix code, but not an overlap-free language[10].*

[10]Note that overlap-freeness alone does not imply unique decodability.

We focus on the overlap relation either in the form ω_{ol} or the form of $\overline{\omega_{\mathrm{ol}}}$. For decompositions the words which do not contain aba or bab are violation-free. For decodings, any word not in $C^*\{abab, baba\}C^*$ is violation-free.

Note that every non-empty word $q \notin C^+$ is violation-free for decodings with respect to C.

In general there is a pattern: For decompositions $q \notin \Sigma^*\mathsf{word}(\mathfrak{v})\Sigma^*$ is violation-free. For decodings $q \notin C^*\mathsf{word}(\mathfrak{v})C^*$ is violation-free.

When two relations interact, as in the case of solid codes, for violation-freeness the corresponding property seems not to be just a simple Boolean junction of the basic properties; this seems to require an expression of the co-locality of the respective defining situations. Neither Definition 3.4 based on Head's work nor our Definition 3.6 covers this adequately. We hope to look at this issue in a subsequent study.

Instead of violating instances one can also consider occurrences of words which, taken together, violate the condition in question although their occurrences may be "unrelated". To this end we modify Definition 3.1 following the pattern of Definition 3.6. In contrast to the violating instances, we consider a property P_ω which is given by an k-ary relation $\omega \subseteq k\text{-}\mathsf{tuples}(\Sigma^+)$. For example: for prefix-freeness we have $(u,v) \in \omega_{\mathrm{p}}^{\neq}$ if $u <_{\mathrm{p}} v$; for overlap-freeness we have $(u,v) \in \omega_{\mathrm{ol}}$ if there exist $w_1, w_1, w_3 \in \Sigma^+$ such that $u = w_1w_2$ and $v = w_2w_3$; and for the intercode property of index n, we have

$$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\mathrm{inter}_n}$$

if there exist $x, y \in \Sigma^+$ such that $v_1v_2\cdots v_{n+1} = xu_1u_2\cdots u_ny$. As our definition covers the overlap relation and a word can have a non-trivial overlap with itself, like $(xyx, xyx) \in \omega_{\mathrm{ol}}$, we cannot assume that the relation ω is irreflexive – if it is binary – in general. On the other hand, all binary relations ω_ϱ with $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$ are irreflexive. In order to make the following definition as general as possible, we let ω be an arbitrary subset of $\mathsf{all\text{-}tuples}(\Sigma^+)$ rather than a k-ary relation.

Definition 3.7. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$, and $\omega \subseteq \mathsf{all\text{-}tuples}(\Sigma^+)$. Let P_ω be the predicate defining ω.*

1. *The word q is said to be P_ω-*admissible for decompositions with respect *to C, if, for all $\mathfrak{u} = (u_1, u_2, \ldots, u_n) \in \omega \cap \mathsf{all\text{-}tuples}(C)$, there exists (at least) one index $1 \le i \le n$ such that $u_i \not\le_{\mathrm{i}} q$.*

2. *The word q is said to be P_ω-*admissible for decodings with respect to C, *if, for all $\mathfrak{u} = (u_1, u_2, \ldots, u_n) \in \omega \cap \mathsf{all\text{-}tuples}(C)$, there exists (at least) one index $1 \le i \le n$ such that there are no C-decodings $q = q_1 u_i q_2$ with $q_1, q_2 \in C^*$.*

Remark 3.2. *For ω_{o} and the relations defining the shuffle hierarchy except $<_{\mathrm{i}}$ the definition of admissibility differs significantly from that of violation-freeness. Consider $u, v \in C$ with $(u, v) \in \omega_{\mathrm{o}}^{\neq}$. The occurrence of v would be a violating instance. However, it is no obstacle to admissibility unless also the word u occurs. This statement holds true for all shuffle relations including $<_{\mathrm{h}}$, but excluding $<_{\mathrm{i}}$.*

For intercodes $\omega_{\mathrm{inter}_n}$, as well as comma-free codes and overlap-freeness, a word q is violation-free if the words in $\mathfrak{u} \in \omega_{\mathrm{inter}_n}$ do not appear in a particular constellation in q as defined by the binary relation $\overline{\omega_{\mathrm{inter}_n}}$. In contrast, for admissibility each word in $\mathfrak{u} \in \omega_{\mathrm{inter}_n}$ is treated individually and can appear anywhere in q.

Example 3.4. *Let $\Sigma = \{a, b\}$.*

1. *Consider the prefix order $<_{\mathrm{p}}$ and the language $C = \{a, ab\}$. The set of words which are admissible for decompositions with respect to C are the words not containing ab, that is, all the words in $a^+ \cup b^+ a^*$; in this case violation-freeness and admissibility coincide because a is an infix of ab. The set of words which are admissible for decodings with respect to C are the words in $\Sigma^+ \setminus (C^* ab C^* \cap C^* a C^*) = \Sigma^+ \setminus C^+ \cup a^+ \cup (ab)^+$.*

2. *For the outfix relation, consider the language $C = \{aa, aba\}$ which is not an outfix code. An admissible word for decompositions must not contain both aba and aa as infixes, that is, must be in $\Sigma^+ \setminus (\Sigma^* aba \Sigma^* \cap \Sigma^* aa \Sigma^*)$. For decodings, such a word must be in $\Sigma^+ \setminus (C^* aba C^* \cap C^* aa C^*)$.*

3. *For the comma-free relation, consider the language $C = \{ab, bba\}$. The language C is not a comma-free code, as $\Sigma^+ C \Sigma^+ \cap C^2 \neq \emptyset$ with ab and $bbabba$ as witnesses. Note that C is a bifix code. For decompositions, a word is admissible if not both, bba and ab, are infixes of this word. For decodings, any word not in $C^* bba C^* \cap C^* ab C^*$ is admissible.*

4. *For solid codes, consider $C = \{ab, ba\}$, which is an infix code, but not an overlap-free language. We focus on the overlap relation in the form*

ω_{ol} rather than $\overline{\omega_{\mathrm{ol}}}$. For decompositions the words which do not contain ab and ba are admissible, that is, all words in $a^+b^* \cup b^*a^+ \cup b^+$. For decodings, any word not in $C^*abC^* \cap C^*baC^*$ is admissible.

The following two theorems show how the different notions of admissibility and violation-freeness are related to each other. The set of relations considered can be divided into two sets with two essentially different behaviours. The first set contains only binary, asymmetric, irreflexive relations and its properties are stated in Theorem 3.1; Figure 1 illustrates the relationships. The remaining properties are covered by Theorem 3.2 below.

$$\begin{tikzcd}
\text{violation-free for decompositions} \arrow[r, shift left, "\text{for all } \varrho \text{ (2)}"] \arrow[d, "\text{for all } \varrho \text{ (1)}", "\text{not conversely}"'] & \text{admissible for decompositions} \arrow[l, shift left, "\text{for } \varrho \in \{\mathrm{p,s,pi,si,b,i}\}, \text{ but not for } \varrho \in \{\mathrm{o,h}\} \text{ (5)}"] \arrow[d, "\text{for all } \varrho \text{ (4)}", "\text{not conversely}"'] \\
\text{violation-free for decodings} \arrow[r, "\text{for all } \varrho \text{ (3)}", "\text{not conversely}"'] & \text{admissible for decodings}
\end{tikzcd}$$

Figure 1: Relation described in Theorem 3.1: The numbers on the arrows refer to the statements in Theorem 3.1. This figure is restricted to $\varrho \in \{\mathrm{p, s, pi, si, b, i, o, h}\}$.

Theorem 3.1. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$ and $\varrho \in \{\mathrm{p, s, pi, si, b, i, o, h}\}$. The following statements hold true:*

1. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-violation-free for decodings, but not conversely.*
2. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-admissible for decompositions with respect to C.*
3. *If the word q is P_ϱ-violation-free for decodings with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*

4. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*

5. *For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}$ the converse of (2) holds true. However, for $\varrho \in \{\mathrm{o}, \mathrm{h}\}$ the converse of (2) does not hold.*

Proof: Since $\omega_\varrho^{\neq}$ is binary and irreflexive, we consider two distinct words u and v such that $u \, \omega_\varrho^{\neq} \, v$. The words u and v are fixed throughout this proof.

Assume that the word q is P_ϱ-violation-free for decompositions with respect to C, that is, $v \not\leq_{\mathrm{i}} q$. In particular, v is not an infix of a decoding of q with respect to C.

Conversely, consider the language $C = \{ab, abab, aa, ba\}$ and note that $abab$ is the sole violating instance of P_ϱ for all relations considered here. The word $aababa$ with the unique C-decoding $(3, (aa, ba, ba))$ is P_ϱ-violation-free for decodings with respect to C, but it is not P_ϱ-violation-free for decompositions with respect to C. This proves (1).

Again, let q be P_ϱ-violation-free for decompositions with respect to C. As $v \not\leq_{\mathrm{i}} q$, trivially v and u cannot both be infixes of q. This proves (2).

Assume that the word q is P_ϱ-violation-free for decodings with respect to C. Then v is not an infix of a decoding of q with respect to C. Thus, trivially v and u cannot both be infixes of a decoding of q either.

Conversely, consider the language $C = \{ab, abbab\}$ and note that $ab \, \omega_\varrho^{\neq} \, abbab$ for all relations considered here. The word $abbab$ with the unique C-decoding $(1, (abbab))$ is not P_ϱ-violation-free for decodings with respect to C, but it is P_ϱ-admissible for decodings with respect to C. This proves (3).

Assume that the word q is P_ϱ-admissible for decompositions with respect to C. Then u and v are not both infixes of q. Hence, they are not both infixes of decodings of q with respect to C.

Conversely, consider the language $C = \{ab, abbab\}$ again, and note that $abbab$ is P_ϱ-admissible for decodings with respect to C, but it is not P_ϱ-admissible for decompositions with respect to C. This proves (4).

For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}$, if q is P_ϱ-admissible for decompositions with respect to C, then $v \not\leq_{\mathrm{i}} q$ because $u <_{\mathrm{i}} v$ due to the choice of ϱ. Hence, q is P_ϱ-violation-free for decompositions with respect to C.

Now, consider $C = \{aa, aba\}$, which is not an outfix code. The word aba contains aba, but not aa. Therefore, aba is P_{o}-admissible and P_{h}-admissible for decompositions with respect to C. On the other hand, the occurrence of aba is a P_{o}-violating and P_{h}-violating instance. This proves (5). $\square$

The situation for overlap-free languages, solid codes, intercodes, and comma-free codes is different from the code properties that are covered by Theorem 3.1; Figure 2 illustrates the relationships stated in Theorem 3.2.

Theorem 3.2. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$ and $\varrho \in \{\text{ol}, \text{solid}, \text{inter}_n, \text{comma-free}\}$. The following statements hold true:*

1. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-violation-free for decodings, but not conversely.*

2. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-violation-free for decompositions with respect to C, but not conversely.*

3. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*

4. *If q is P_ϱ-asmissible for decodings with respect to C, this does not imply that q is P_ϱ-violation-free for decodings or decompositions with respect to C. If q is P_ϱ-violation-free for decodings or decompositions with respect to C, this does not imply that q is P_ϱ-admissible for decodings with respect to C.*

5. *If $q \in C^+$ and q is P_{solid}-violation-free for decodings with respect to C, then q is also P_{solid}-violation-free for decompositions with respect to C.*

Proof: Let $\varrho \in \{\text{ol}, \text{inter}_n\}$ and let $\mathfrak{u}, \mathfrak{v}$ be word tuples such that $\mathfrak{u}\ \overline{\omega_\varrho}\ \mathfrak{v}$. For $\varrho = \text{inter}_n$, we require that $\mathfrak{u}, \mathfrak{v} \in \textsf{all-tuples}(C)$ and we let $\mathfrak{w} = \mathfrak{u} \cdot \mathfrak{v}$ be the concatenation of the tuples $\mathfrak{u}$ and $\mathfrak{v}$. For $\varrho = \text{ol}$, we only require that $\mathfrak{u} \in \textsf{all-tuples}(C)$ and we let $\mathfrak{w} = \mathfrak{u}$. In both cases, we have $\mathfrak{w} \in \omega_\varrho \cap \textsf{all-tuples}(C)$.

The case $P_{\text{comma-free}}$ is covered as a special case of P_{inter_n}, whereas the case $P_{\text{solid}} = P_{\text{i}} \wedge P_{\text{ol}}$ requires special attention. Note that the positive statements of (1,2,3) follow for P_{solid} because they hold for P_{ol} (proven below) and P_{i} (Theorem 3.1).

If q is P_ϱ-violation-free for decompositions with respect to C, then $\textsf{word}(\mathfrak{v})$ is not a proper infix of q. In particular, $\textsf{word}(\mathfrak{v})$ is not an infix of a decoding of q as described in Definition 3.6. Hence, q is P_ϱ-violation-free for decodings with respect to C. The same property holds for P_{solid} because it holds for P_{ol} and P_{i} by Theorem 3.1.

$$\begin{tikzcd}
\text{violation-free for decompositions} \arrow[d, shift left=2, "\text{for all } \varrho \text{ (1)}"] & & \text{admissible for decompositions} \arrow[ll, "\text{for all } \varrho \text{ (2)}"', "\text{not conversely}"] \arrow[d, "\text{for all } \varrho \text{ (3)}", "\text{not conversely}"'] \\
\text{violation-free for decodings} \arrow[u, shift left=2, "\text{only if } \varrho = \text{solid and } q \in C^+ \text{ (5)}"] \arrow[rr, dashed, no head, "\text{incomparable (4)}"] & & \text{admissible for decodings}
\end{tikzcd}$$

Figure 2: Relation described in Theorem 3.2: The numbers on the arrows refer to the statements in Theorem 3.2. This figure is restricted to $\varrho \in \{\text{ol}, \text{solid}, \text{inter}_n, \text{comma-free}\}$.

Conversely, for P_{inter_n} with $n \geq 1$, let $C = \{ab, ba, a(ab)^{n+1}\}$. The word $a(ab)^{n+1}$, with the unique C-decoding $(1, (a(ab)^{n+1}))$, is P_{inter_n}-violation-free for decodings with respect to C, but it is not P_{inter_n}-violation-free for decompositions with respect to C as it contains the violating instance $(ab)^{n+1}$. For P_{ol}, let $C = \{ab, ba, abaa\}$. The word $abaa$, with the unique C-decoding $(1, (abaa))$, is P_{ol}-violation-free for decodings with respect to C, but it is not P_{ol}-violation-free for decompositions with respect to C as it contains the violating instance aba. For P_{solid}, this result can only be obtained if q does not have a C-decoding and is, therefore, P_{solid}-violation-free for decodings with respect to C, but contains a P_{solid}-violating instance as infix; otherwise, statement (5) would be contradicted. This proves (1).

Assume q is P_ϱ-admissible for decompositions with respect to C. Hence, not all of the words in $\mathfrak{w}$ appear as infixes of q. Because $\mathsf{word}(\mathfrak{v})$ contains all words from $\mathfrak{w}$ as infixes, we have $\mathsf{word}(\mathfrak{v}) \not\leq_{\text{i}} q$. Hence, q is P_ϱ-violation-free for decompositions with respect to C. This proves the "forward direction" of (2).

If q is P_ϱ-admissible for decompositions with respect to C, then not all of the words from $\mathfrak{w}$ can be infixes of q. In particular, they cannot all be infixes of decodings of q. Hence, q is P_ϱ-admissible for decodings with respect to C. This proves the "forward direction" of (3).

For $\varrho = \text{inter}_n$ with $n \geq 1$, let $C = \{ab, ba\}$. The word $(ab)^{n+1}$ is P_{inter_n}-violation-free for decodings and decompositions with respect C, but it is P_{inter_n}-admissible for decodings with respect to C since ba does not appear in the unique C-decoding $(n+1, (ab, \ldots, ab))$. Furthermore, $(ab)^{n+1}$ is not P_{inter_n}-admissible for decompositions with respect to C. On the other hand, the word $abba$ with the unique decoding $(2, (ab, ba)$ is P_{inter_n}-violation-free for decodings and decompositions with respect to C, but it is not P_{inter_n}-admissible for decodings or decompositions with respect to C. Note that we do not require that ab or ba appears in n or $n+1$ distinct positions in $abba$. This proves (4) and the "converse directions" of (2) and (3) do not hold for intercodes of index n.

For $\varrho \in \{\text{ol}, \text{solid}\}$, let $C = \{abb, bab\}$. The word $abbabb$ is P_ϱ-admissible for decodings with respect to C, because bab does not occur in a decoding of $abbabb$. However, $abbabb$ contains the violating instance $abbab$ as described in Definition 3.6. Therefore, $abbabb$ is not P_ϱ-violation-free for decodings or decompositions with respect to C. Furthermore, $abbabb$ is not P_ϱ-admissible for decompositions with respect to C. The word $abbbab$, on the other hand, is P_ϱ-violation-free for decodings or decompositions with respect to C, but it is not P_ϱ-admissible for decodings or decompositions with respect to C. This proves (4) and the "converse directions" of (2) and (3) do not hold.

Let $q = u_1 u_2 \cdots u_n$ with $u_1, u_2, \ldots, u_n \in C$ be P_{solid}-violation-free for decodings with respect to C. Suppose q contains a P_{solid}-violating instance v as infix If v it is a violating instance of P_{i}, let $w = v \in C$; if v it is a violating instance of P_{ol}, let $w <_{\text{p}} v$ such that $w \in C$ and there exists $w' <_{\text{s}} v$ such that $w' \in C$ and $|ww'| > |v|$. We distinguish five cases:

- If $w = u_i$, for some $1 \leq i \leq n$, such that $v \leq_{\text{p}} w u_{i+1} \cdots u_n$ then v would be a witness that q is not P_{solid}-violation-free for decodings with respect to C.

- If w was an infix of any u_i, then u_i would be a witness that q is not P_{i}-violation-free for decodings with respect to C.

- If $w = u_i u_{i+1} \cdots u_j$ for some i, j with $1 \leq i < j \leq n$, then w (in the decoding $q = u_1 \cdots u_{i-1} w u_{j+1} \cdots u_n$) would be a witness that q is not P_{i}-violation-free for decodings with respect to C.

- If there existed $1 \leq i < n$ and $x, y, z \in \Sigma^+$ such that $u_i = xy$, $w = yz$ and $w \leq_{\text{p}} u_{i+1} \cdots u_n$, then xyz is a witness that q is not P_{ol}-violation-free for decodings with respect to C.

- If there existed $1 < i \leq n$ and $x, y, z \in \Sigma^+$ such that $w = xy$, $u_i = yz$ and $w \leq_{\mathrm{s}} u_1 \cdots u_{i-1}$, then xyz is a witness that q is not P_{ol}-violation-free for decodings with respect to C.

This covers all possibilities of how w, as prefix of v, can be located in q. This proves (5). □

One can intuit the relativization of a code property P as follows: If a word $q \in C^+$ satisfies the relativized property with respect to C, then the word should be uniquely decodable over C. As we show next, this intuition holds true for the notion of admissibility, but does so only for some special properties when considering violation-freeness. However, the converse of this statement is not true: If a word q is uniquely decodable over C, then q is not necessarily P-violation-free or P-admissible for decompositions or decodings with respect to C. For example, consider the prefix-free property and $C = \{ab, aba\}$. The word $q = ababa$ has the unique C-decoding $(2, (ab, aba))$; it is, however, neither P_{p}-violation-free nor P_{p}-admissible for decompositions or decodings with repect to C.

Theorem 3.3. *Let $\omega \subseteq k\text{-}\mathsf{tuples}(\Sigma^+)$ be a k-ary relation such that if a non-empty language D satisfies P_ω, then D is a code (all words over Σ^+ have at most one D-decoding). Let $C \subseteq \Sigma^+$ be a non-empty language. If q is P_ω-admissible for decodings or decompositions, then q has at most one C-decoding.*

Proof: Suppose $q \in \Sigma^+$ has two C-decodings and is P_ω-admissible for decodings or decompositions relative to C. Let

$$(m, (u_1, u_2, \ldots, u_m)) \text{ and } (n, (v_1, v_2, \ldots, v_n))$$

be two distinct C-decodings of q. Let $C' = \{u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n\} \subseteq C$. As q is P_ω-admissible for decodings or decompositions with respect to C, for all $\mathfrak{w} \in k\text{-}\mathsf{tuples}(C')$ we have $\mathfrak{w} \notin \omega$. Therefore, C' satisfies the property P_ω and must be a code. However, the word q has two C'-decodings – a contradiction! □

This result easily extends to violation-free words for the properties P_ϱ with $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}\}$, using Theorem 3.1. For P_{solid} we need to observe that P_{solid}-violation-freeness with respect to a language C implies P_{i}-violation-freeness with respect to C.

Corollary 3.1. *Let $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}\}$ and $C \subseteq \Sigma^+$ be a non-empty language. If q is P_ϱ-violation-free for decodings or decompositions, then q has at most one C-decoding.*

A similar result cannot be obtained for intercodes or comma-free codes: Let $C = \{ab, abab\}$ and $n \geq 1$ The word $abab$ clearly has two distinct C-decodings. However, $abab$ is P_{inter_n}-violation-free for decodings or decompositions with respect to C. Indeed, for comma-freeness the shortest violating instance over C is $(ab)\ \omega_{\text{comma-free}}\ (ab, abab)$ or $(ab)\ \omega_{\text{comma-free}}\ (abab, ab)$.

3.4 Relativized Codes

We have arrived at four notions of how a word may satisfy the predicate P_ϱ for a given non-empty language $C \subseteq \Sigma^+$:

1. vf-decomp: Violation-free for decompositions;
2. vf-decod: Violation-free for decodings;
3. adm-decomp: Admissible for decompositions;
4. adm-decod: Admissible for decodings.

Let $\mathfrak{M}$ be the set of these notions. Each $\mu \in \mathfrak{M}$ gives rise to a definition of a class of relativized codes as follows:

Definition 3.8. *Let C and L be non-empty subsets of Σ^+, let $\mu \in \mathfrak{M}$ and let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}.$$

The language C is a P_ϱ-μ code relative to L if every word in L has the property P_ϱ-μ with respect to C.

Let $\mathcal{C}^\mu_\varrho(L)$ be the class of P_ϱ-μ codes relative to L. Let $\mathcal{L}^\mu_\varrho(C)$ be the class of non-empty languages $L \subseteq \Sigma^+$ such that C is a P_ϱ-μ code relative to L. The following statements are consequences of the results obtained so far.

Theorem 3.4. *In the statements below the symbols μ, ϱ, C, and L are defined as follows: $\mu \in \mathfrak{M}$,*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\},$$

$C, L \subseteq \Sigma^+$, $C \neq \emptyset$, $L \neq \emptyset$. The following statements hold true:

1. *For all μ, ϱ and C, the set $\mathcal{L}^{\mu}_{\varrho}(C)$ is closed under arbitrary unions. Therefore, it contains a unique maximal language denoted $M^{\mu}_{C,\varrho}$.*
2. *For all μ, ϱ and L, the set $\mathcal{C}^{\mu}_{\varrho}(L)$ is closed under non-empty intersections.*
3. $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{vf-decod}}_{\varrho}(L)$ *and* $\mathcal{C}^{\text{adm-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decod}}_{\varrho}(L)$. *The inclusions are proper for some L.*
4. $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) = \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}$; $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\mathrm{o}, \mathrm{h}\}$; *and* $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \supseteq \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}$. *The inclusions are proper for some L.*
5. $\mathcal{C}^{\text{vf-decod}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decod}}_{\varrho}(L)$ *for* $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$. *The inclusion is proper for some L.*
6. *If $C' \subseteq C$, $C' \neq \emptyset$, then $\mathcal{L}^{\mu}_{\varrho}(C) \subseteq \mathcal{L}^{\mu}_{\varrho}(C')$ for all μ and ϱ.*
7. *If $L' \subseteq L$ then $\mathcal{C}^{\mu}_{\varrho}(L) \subseteq \mathcal{C}^{\mu}_{\varrho}(L')$ for all μ and ϱ.*

Proof: All statements are easy consequences of the definitions and of Theorems 3.1 and 3.2. $\square$

Theorem 3.4 summarizes simple aspects of relativizing code properties. More detailed issues can be learned from Theorems 3.1 and 3.2. In both cases, the statements are limited to specific code properties ϱ. To identify the common scheme for a wider class of code properties is still an open problem.

3.5 The Old and New Definitions Compared

We outline how the definitions of code relativization given in [2] and [9] compare to the ones in the present paper. While we attempted to maintain consistency, it was inevitable that some definitions would change given the fact that a detailed look prompted by [13] revealed the need for a more general and less uniform approach. Hence, when reading the older papers together with this one, it is important to watch for slight, but possibly important, differences in the definitions, before using the statements of theorems. Particular attention needs to be paid to the issues in the following remark.

Remark 3.3. *Let $L, C \subseteq \Sigma^+$ be non-empty languages.*

1. *Let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. Note that a subset ω of* $\textsf{1-tuples}(\Sigma^+) \cup \textsf{2-tuples}(\Sigma^+)$ *describes the predicate P, that is $P = P_\omega$, if and only if*

$$\begin{aligned} P(\{x,y\}) = 0 &\iff (x,y) \in \omega \text{ or } (y,x) \in \omega \text{ and} \\ P(\{x\}) = 0 &\iff (x) \in \omega \text{ or } (x,x) \in \omega. \end{aligned}$$

   Let $P = P_\omega$ be described by a set of tuples ω. A word $q \in C^+$ is P-admissible for C in the sense of Definition 3.1 if and only if it is P-admissible for decodings with respect to C in the sense of Definition 3.7. Furthermore, if $L \subseteq C^+$, then C is a P-code relative to L in the sense of Definition 3.3 if and only if C is a P-admissible code for decodings relative to L in the sense of Definition 3.8.

   Note that, Definitions 3.1 and 3.3 do not cover the cases when $q \notin C^+$ and $L \not\subseteq C^+$, respectively, while, in this paper, we naturally extend these definitions to all words $q \in \Sigma^+$ and languages $L \subseteq \Sigma^+$.

2. *C is a solid code relative to L according to Definition 3.4 if and only if every word $q \in L$ is P_{solid}-violation-free for decompositions with respect to C in the sense of Definition 3.6 or, equivalently, if C is a P_{solid}-violation-free code for decompositions relative to L in the sense of Definition 3.8.*

We suggest that the framework of this paper supersede those of [2, 9]. The concepts are still not ideal, but approaching what we consider the right ones.

4 Decidability Questions

In general, given non-empty languages $C, L \subseteq \Sigma^+$ and a code property P, one would like to know whether C is a P-code relative to L. In practical terms, we are given a language L of messages to be transmitted. We are also given a target for the transmission quality expressed by the predicate P. We want to know whether a given candidate code C serves the purpose. This gives rise to decidability problems for relativized codes.

For unrelativized codes, that is codes relative to Σ^+, results regarding the decidability of code properties as of 1996 are proved or summarized in [15, 14]. Further details are found in [3] and [4]. It is known that code properties are usually decidable when C is a regular language, and undecidable when C is a linear language. In [4] it is shown that the boundary between decidability and undecidability is significantly lower than that of linear languages. For relativized code properties this implies that one should not expect decidability unless C is regular. Regarding assumptions about L, we only consider the case of L being regular as well. At present we do not know to which extent this restriction can be lifted.

We first review two notions which we use in some of the proofs in this section.

1. Let $L \subseteq \Sigma^+$. The *syntactic congruence* $\sim_L$ with respect to L is defined as follows: For $u, v \in \Sigma^+$, $u \sim_L v$ if and only if, for all $x, y \in \Sigma^*$, either xuy and xvy are both in L or both not in L. The *syntactic semigroup* of L is the quotient semigroup $\Sigma^+/{\sim_L}$. Each element of the syntactic semigroup of L is a *syntactic class* which can be viewed as a language itself. For a word u we write $[u]_L$ to denote its syntactic class. The syntactic semigroup of a language L is finite if and only if L is regular. For languages L_1 and L_2 over the same alphabet Σ, $\sim_{(L_1,L_2)}$ denotes the intersection of the congruences $\sim_{L_1}$ and $\sim_{L_2}$. For additional basic information on syntactic semigroups we refer to [17, 24].

2. The second notion to consider is that of *shuffling on a trajectory.* This concept is widely used in order to describe code properties [29, 18, 19, 20, 21, 14]. A *trajectory* t is a word over the alphabet $\{0, 1\}$. The result of shuffling two words u and v on the trajectory t is a word $w = u \sqcup\!\sqcup_t v$ that is obtained by using all letters from u and v where the trajectory t determines in which places to use letters from u or v. Shuffling is defined recursively by

$$\lambda \sqcup\!\sqcup_\lambda \lambda = \lambda, \quad au \sqcup\!\sqcup_{0t} v = a(u \sqcup\!\sqcup_t v), \quad u \sqcup\!\sqcup_{1t} bv = b(u \sqcup\!\sqcup_t v)$$

   where $a, b \in \Sigma$, $u, v \in \Sigma^*$, and $t \in \{0, 1\}^*$. Note that $u \sqcup\!\sqcup_t v$ is only defined if $|u| = |t|_0$ and $|v| = |t|_1$. This concept is extended to languages L_1, L_2 and a set of trajectories $\mathbf{t}$ by

$$L_1 \sqcup\!\sqcup_{\mathbf{t}} L_2 = \{u \sqcup\!\sqcup_t v \mid u \in L_1, v \in L_2, t \in \mathbf{t}\}.$$

   The shuffle of two regular languages on a regular set of trajectories yields a regular language.

Let P_ϱ be a code property and $C \subseteq \Sigma^+$ be a non-empty language. For each of the four notions of relativized codes $\mu \in \mathfrak{M}$ there is a maximal language $M^{\mu}_{C,\varrho}$ such that a language L is a P_ϱ-μ code relative to C if and only if $L \subseteq M^{\mu}_{C,\varrho}$, as stated in Theorem 3.4.

We show that $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ are effectively constructible regular languages for all P_ϱ considered in this paper. Thus, to decide whether or not a given regular language is a P_ϱ-violation-free code for decompositions or decodings relative to another regular language, one can test for inclusion of regular languages. Let $V_{C,\varrho} = \{v \in C \mid v \text{ is a violating instance of } P_\varrho \text{ in } C\}$. Here, a *violating instance* is a $\mathsf{word}(\mathfrak{v})$ as used in Definition 3.6. Since P_{ol}-violation-freeness and P_{solid}-violation-freeness, do not follow the general definition, these properties are treated separately.

Lemma 4.1. *Let $C \subseteq \Sigma^+$ be a non-empty language and let*

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{inter}_n, \text{comma-free}\}.$$

We have $M^{\text{vf-decomp}}_{C,\varrho} = \Sigma^+ \setminus \Sigma^* V_{C,\varrho} \Sigma^*$ and $M^{\text{vf-decod}}_{C,\varrho} = \Sigma^+ \setminus C^* V_{C,\varrho} C^*$.

Proof: The languages $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ contain precisely those words which are violation-free for decompositions or decodings, respectively, with respect to C. This is a direct consequence of Definition 3.6. □

Theorem 4.1. *For*

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{ol}, \text{solid}, \text{inter}_n, \text{comma-free}\}$$

and regular $C \subseteq \Sigma^+$ the languages $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ are effectively regular.

Proof: The sets of violations can be expressed as $V_{C,\text{p}} = C \cap C\Sigma^+$, $V_{C,\text{s}} = C \cap \Sigma^+ C$, $V_{C,\text{pi}} = C \cap \Sigma^* C \Sigma^+$, $V_{C,\text{si}} = C \cap \Sigma^+ C \Sigma^*$, $V_{C,\text{b}} = V_{C,\text{p}} \cup V_{C,\text{s}}$, $V_{C,\text{i}} = C \cap (\Sigma^+ C \Sigma^* \cup C\Sigma^+)$, $V_{C,\text{inter}_n} = C^{n+1} \cap \Sigma^+ C^n \Sigma^+$, $V_{C,\text{comma-free}} = CC \cap \Sigma^+ C \Sigma^+$, which are all regular languages. The sets of violations for outfix-codes and hypercodes can be expressed using shuffling on a trajectory: we have $V_{C,\text{h}} = C \cap (C \sqcup_{\mathbf{t}_\text{h}} \Sigma^+)$ for $\mathbf{t}_\text{h} = \{0,1\}^+$ and $V_{C,\text{o}} = C \cap (C \sqcup_{\mathbf{t}_\text{o}} \Sigma^+)$ for $\mathbf{t}_\text{o} = 0^* 1^+ 0^*$; both sets of violations are regular.

Using Lemma 4.1, we obtain that $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ are regular because $V_{C,\varrho}$ is regular for $\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{inter}_n, \text{comma-free}\}$. All steps in these constructions are effective.

For $\varrho = \mathrm{ol}$, the set of violations can be written as

$$V_{C,\mathrm{ol}} = \{xyz \mid x,y,z \in \Sigma^+, xy, yz \in C\} = \bigcup_{\substack{X,Y,Z \in \Sigma^+/\sim_C \\ XY \subseteq C, YZ \subseteq C}} XYZ$$

which is an effectively regular language. As before, we obtain $M_{C,\mathrm{ol}}^{\text{vf-decomp}} = \Sigma^+ \setminus \Sigma^* V_{C,\mathrm{ol}} \Sigma^*$.

The set $M_{C,\mathrm{ol}}^{\text{vf-decod}}$ cannot be expressed in a similar manner as above; nevertheless, it is effectively regular, given as

$$M_{C,\mathrm{ol}}^{\text{vf-decod}} = \Sigma^+ \setminus \bigcup_{\substack{X,Y,Z \in \Sigma^+/\sim_C \\ XY \subseteq C, YZ \subseteq C}} \big(C^* XY(Z\Sigma^* \cap C^+) \cup (C^+ \cap \Sigma^* X)YZC^*\big)\,.$$

Finally, we have $M_{C,\mathrm{solid}}^{\text{vf-decomp}} = M_{C,\mathrm{i}}^{\text{vf-decomp}} \cap M_{C,\mathrm{ol}}^{\text{vf-decomp}}$ and $M_{C,\mathrm{solid}}^{\text{vf-decod}} = M_{C,\mathrm{i}}^{\text{vf-decod}} \cap M_{C,\mathrm{ol}}^{\text{vf-decod}}$. $\square$

Since the constructions of the regular languages in both Lemma 4.1 and Theorem 4.1 are effective, one concludes:

Corollary 4.1. *Let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}.$$

For given regular languages C and L it is decidable

1. *whether or not C is a P_ϱ-violation-free code for decompositions relative to L, and*

2. *whether or not C is a P_ϱ-violation-free code for decodings relative to L.*

This result can be extended to P_ϱ-admissible codes for decompositions for those properties for which P_ϱ-admissibility and P_ϱ-violation-freeness coincide – see Theorem 3.1.

Corollary 4.2. *Let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}.$$

For given regular languages C and L it is decidable whether or not C is an P_ϱ-admissible code for decompositions relative to L.

The situation changes when considering admissibility for decodings. Decidability cannot be expressed as an inclusion test of two regular languages as before.

Proposition 4.1. *For a given regular $C \subseteq \Sigma^*$ the language $M_{C,p}^{\text{adm-decod}}$ is not necessarily regular.*

Proof: Let $\Sigma = \{0, 1\}$ and $C = 10^*$. Obviously, a word $w \in 10^i10^j \in C^2$ belongs to $M_{C,\varrho}^{\text{adm-decod}}$ if and only if $i = j$. Therefore, the language $M_{C,\varrho}^{\text{adm-decod}}$ cannot be regular as its intersection with the regular language C^2 is not regular. □

Deciding whether or not a regular language C is an P_ϱ-admissible code for decodings relative to a regular language L works in two stages: first, decide whether or not C is a code relative to L, that is, every word in L has at most one C-decoding; then, verify that every decoding of a word in L is P_ϱ-admissible. We focus only on code properties P_ϱ defined by irreflexive binary relations; this excludes solid codes, intercodes, comma-free codes and overlap-free languages.

The next lemma forms the basis for deciding whether or not a regular language C is a code relative to a regular language L. The lemma itself does not require that the languages L and C be regular.

Lemma 4.2. *Let $L, C \subseteq \Sigma^+$ be non-empty languages. The language C is a code relative to L if and only if, for all syntactic classes $Y \in \Sigma^+/{\sim_C}$ contained in C, one has*

$$(C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^* = \emptyset.$$

Proof: Suppose C is not a code relative to L. There exists a word $w = u_1 \cdots u_n = v_1 \cdots v_m \in L$ such that $u_1, \ldots, u_n, v_1, \ldots, v_m \in C$ and $u_i \neq v_i$ for some $1 \leq i \leq \min\{n, m\}$. Let i be minimal such that $u_i \neq v_i$ and, by symmetry, assume that $u_i <_p v_i$. Let $Y = [u_i]_C$ and observe that $z = u_{i+1} \cdots u_n$ belongs to $(C^*Y)^{-1}L$ as well as C^*; furthermore, since $u_i^{-1}v_i \in Y^{-1}C \setminus \{\lambda\}$, we obtain $z = (u_i^{-1}v_i)v_{i+1} \cdots v_m \in (Y^{-1}C \setminus \{\lambda\})C^*$. Therefore,

$$(C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^*$$

is not empty.

Conversely, suppose that

$$z \in (C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^*$$

exists for some $Y \in \Sigma^+/{\sim_C}$ with $Y \subseteq C$. Let $x_1, \ldots, x_i \in C^*$ and $y \in Y$ such that $w = x_1 \cdots x_i yz \in L$. There are $u_1, \ldots, u_n \in C$ such that $z = u_1 \cdots u_n$. Furthermore, we let $v_0 \in y(Y^{-1}C \setminus \{\lambda\})$ and $v_1, \ldots, v_m \in C$ such that $yz = v_0, \ldots, v_m$; note that $v_0 \in C$. Thus, we found two factorizations

$$w = x_1 \cdots x_i y u_1 \cdots u_n = x_1 \cdots x_i v_0 \cdots v_m$$

of a word that belongs to L where all factors belong to C and $y \neq v_0$. We conclude that C is not a code relative to L. □

Theorem 3.3 and Lemma 4.2 lead to a general method for deciding whether a regular language C is an P_ϱ-admissible code for decodings relative to a regular language L provided the relation ω_ϱ is binary and recognizable in a transducer model with a decidable emptiness or membership problem. This applies to

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}.$$

In [15] it is shown that the emptiness problem for a transducer model is decidable if and only if its membership problem is decidable. Furthermore, if the emptiness problem of a transducer machine recognizing ω_ϱ is decidable, then for regular X, Y it is decidable whether or not $x \in X$ and $y \in Y$ exists such that $x \, \omega_\varrho \, y$.

Theorem 4.2. *Let $C, L \subseteq \Sigma^+$ be non-empty regular languages and let P_ϱ be a code property such that ω_ϱ is irreflexive and recognizable by a transducer machine with decidable emptiness problem. It is decidable whether or not C is a P_ϱ-admissible code for decodings relative to L.*

Proof: According to Theorem 3.3, for C to be an P_ϱ-admissible code for decodings relative to L, it is necessary that C is a code relative to L. By Lemma 4.2, we can decide whether or not C is a code relative to L by performing a series of emptiness test of regular languages. Henceforth, we assume that we have preformed this test and that C is a code relative to L.

We will show that, under the premise that C is a code relative to L, C is a P_ϱ-admissible code for decodings relative to L if and only if for all syntactic classes $X, Y \in \Sigma^+/\sim_{(C,L)}$ such that $X, Y \subseteq C$ and there exist $x \in X$ and $y \in Y$ with $x \; \omega_\varrho \; y$ or $y \; \omega_\varrho \; x$, we have

$$C^*XC^*YC^* \cap L = \emptyset.$$

Recall that one can decide whether or not there are $x \in X$ and $y \in Y$ such that $x \; \omega_\varrho \; y$ or $y \; \omega_\varrho \; x$ because ω_ϱ is recognizable by a transducer machine with decidable emptiness problem [15].

Now, suppose that $w \in C^*XC^*YC^* \cap L$ exists for a pair $X, Y \in \Sigma^+/\sim_{(C,L)}$ such that $X, Y \subseteq C$ and there exist $x \in X$ and $y \in Y$ with $x \; \omega_\varrho \; y$ or $y \; \omega_\varrho \; x$. Let $w \in u_1Xu_2Yu_3$ for $u_1, u_2, u_3 \in C^*$. One obtains that $u_1Xu_2Yu_3 \subseteq L$ and, therefore, $u_1xu_2yu_3 \in C^*XC^*YC^* \cap L$ with $x \; \omega_\varrho \; y$ or $y \; \omega_\varrho \; x$. Hence C is not a P_ϱ-admissible code for decodings relative to L.

Conversely, let $w \in L$ be a witness for the fact that C is not a P_ϱ-admissible code for decodings relative to L; that is, two words $x, y \in C$ such that $x \; \omega_\varrho \; y$ or $y \; \omega_\varrho \; x$ appear in decodings of w over C; note that we cannot have $x = y$ since ω_ϱ is irreflexive. As C is a code relative to L, x and y appear in the same decoding; thus, without loss of generality, we can factorize $w = u_1xu_2yu_3$ with $u_1, u_2, u_3 \in C^*$ and $C^*[x]_{(C,L)}C^*[y]_{(C,L)}C^* \cap L$ cannot be empty. $\square$

With the tools used in this section we cannot answer the following questions:

1. For $\varrho \in \{\mathrm{o}, \mathrm{h}, \mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$ and given regular languages C, L, is it decidable whether or not C is a P_ϱ-admissible code for decompositions relative to L?

2. For $\varrho \in \{\mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$ and given regular languages C, L, is it decidable whether or not C is a P_ϱ-admissible code for decodings relative to L?

5 Final Remarks

In information processing, coding serves several purposes. These are expressed by code properties. In a real information transmission system, messages arrive with different probabilities including many with probability 0. Unless data ideal compression is applied, which would essentially eliminate the latter and make all messages equally likely, coding should take into account which messages are likely to be encoded. This idea is modelled by relativized codes. Thus the standard code properties, both information theoretically and in terms of combinatorics, have their relativized counterparts, relativized to the language of likely messages to be encoded. This is very much in the spirit of Shannon's channel coding theorem [26] where messages of probability 0 are practically ignored.

Contrary to what was envisaged in [2], no uniformly acceptable relativization seems possible. Instead, examining various potential models, we arrived at four definitions, each of which seems to be equally well motivated.

We compare these models both among each other and to intuitive expectations. We also consider their decidability properties. We have indicated a few open questions. Many more could have been mentioned.

References

[1] J. Berstel, D. Perrin, C. Reutenauer: *Codes and Automata. Encyclopedia of Mathematics and Its Applications* **129**. Cambridge University Press, Cambridge, 2010.

[2] M. Daley, H. Jürgensen, L. Kari, K. Mahalingam: Relativized codes. *Theoret. Comput. Sci.* **429** (2012), 54–64.

[3] K. Dudzinski, S. Konstantinidis: Formal descriptions of code properties: Decidability, complexity, implementation. *Internat. J. Foundations Comput. Sci.* **23**(1) (2012), 67–85.

[4] H. Fernau, K. Reinhardt, L. Staiger: Decidability of code properties. *RAIRO Inform. Théor. Appl.* **41**(3) (2007), 243–259.

[5] E. R. Gümüştop: *Varieties of Codes and Codifiable Varieties of Monoids.* PhD Dissertation, Binghampton University, State University of New York, 1997.

[6] F. Guzmán: Decipherability of codes. *J. Pure Appl. Algebra* **141** (1999), 13–35.

[7] T. Harju, J. Karhumäki: On the defect theorem and simplifiability. *Semigroup Forum* **33** (1986), 199–217.

[8] T. Head, A. Weber: Deciding multiset decipherability. *IEEE Trans. Inform. Theory* **41** (1995), 291–297.

[9] T. Head: Unique decipherability relative to a language. *Tamkang J. Math.* **11** (1980), 59–66.

[10] T. Head: Deciding the immutability of regular codes and languages und finite transductions. *Inform. Process. Lett.* **31** (1989), 239–241.

[11] T. Head: Relativized code concepts and multi-tube DNA dictionaries. In C. S. Calude, G. Păun (editors): *Finite versus Infinite – Contributions to an Eternal Dilemma.* 175–186. Springer-Verlag, London, 2000.

[12] T. Head. Draft of notes for Form. Lang. Sem. Thurs. Sept. 12, 2002, 2002, 3 pp. Personal Communication.

[13] H. Jürgensen: Markers and deterministic acceptors for non-deterministic languages. *J. of Automata, Languages and Combinatorics* **14** (2009), 33–62.

[14] H. Jürgensen, S. Konstantinidis: Codes. In Rozenberg and Salomaa [25], 1, 511–607.

[15] H. Jürgensen, K. Salomaa, S. Yu: Transducers and the decidability of independence in free monoids. *Theoret. Comput. Sci.* **134** (1994), 107–117.

[16] H. Jürgensen, S. S. Yu: Relations on free monoids, their independent sets, and codes. *Internat. J. Comput. Math.* **40** (1991), 17–46.

[17] G. Lallement: *Semigroups and Combinatorial Applications*. John Wiley & Sons, New York, 1979.

[18] D. Y. Long: k-Outfix codes. *Chinese Ann. Math. Ser. A* **10** (1989), 94–99, in Chinese.

[19] D. Y. Long: k-Prefix codes and k-infix codes. *Acta Math. Sinica* **33** (1990), 414–421, in Chinese.

[20] D. Y. Long: On the structure of some group codes. *Semigroup Forum* **45** (1992), 38–44.

[21] D. Y. Long: k-Bifix codes. *Riv. Mat. Pura Appl.* **15** (1994), 33–55.

[22] D. J. C. MacKay: *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, 6th ed., 2007.

[23] K. Mahalingam: *Involution Codes: with Application to DNA Strand Design.* PhD thesis, University of South Florida, 2004.

[24] J.-E. Pin: Syntactic semigroups. In Rozenberg and Salomaa [25], 1, 679–746.

[25] G. Rozenberg, A. Salomaa (editors): *Handbook of Formal Languages.* Springer-Verlag, Berlin, 1997.

[26] C. E. Shannon: A mathematical theory of communication. *Bell System Tech. J.* **27** (1948), 379–423, 623–656.

[27] H. J. Shyr: *Free Monoids and Languages.* Hon Min Book Company, Taichung, third ed., 2001.

[28] H. J. Shyr, G. Thierrin: Codes and binary relations. In M. P. Malliavin (editor): *Séminaire d'algèbre Paul Dubreil, Paris 1975–1976, (29ème Année). Lecture Notes in Computer Science* **586**, 180–188, Springer-Verlag, Berlin, 1977.

[29] G. Thierrin, S. S. Yu: Shuffle relations and codes. *J. Inform. and Optim. Sci.* **12** (1991), 441–449.

[30] A. Weber, T. Head: The finest homophonic partition and related code concepts. *IEEE Trans. Inform. Theory* **42** (1996), 1569–1575.

[31] S.-S. Yu: *Languages and Codes.* Tsang Hai Book Publishing Co., Taichung, Taiwan, 2005.