What is transcription?

Claus Huitfeldt
Department of Philosophy, University of Bergen, Norway

C. M. Sperberg-McQueen
World Wide Web Consortium / MIT Computer Science and Artificial Intelligence Laboratory

Correspondence:
Claus Huitfeldt,
Department of Philosophy,
University of Bergen,
PO Box 7805, N-5020
Bergen, Norway.
E-mail:
claus.huitfeldt@fof.uib.no

Abstract

This paper describes preliminary sketches for a formal account of transcription as it is performed in scholarly editing and in the creation of digital resources. After a general outline of our approach, we present two formal models of transcription. The first addresses only the very simplest cases, the second addresses some but not all of the gaps in the first. Finally, we mention some less simple cases and discuss some elaborations of the model which we hope to develop in future work.

1 Introduction

What is transcription? This article describes preliminary sketches for a formal account of transcription as it is performed in scholarly editing and in the creation of digital resources.

Sections 2 and 3 describe the problem and the general outlines of our approach. Sections 4 and 5 then present two formal models of transcription, the first addressing only the very simplest cases, the second addressing some but not all of the gaps in the first. Sections 6 and 7 then mention some less simple cases and discuss some elaborations of the model, which we hope to develop in future work.

We believe that a formal account of transcription may provide insight into the nature of electronic representations of cultural artifacts, digital preservation, and text encoding, and that it may have practical consequences for quality assurance and for the representation of markup semantics in formal systems. It may also allow the expression of various conditions on consistency and adequacy of transcriptions, and some account of the ways in which transcriptions of primary source materials can be right or wrong. Because our work is still of a preliminary nature, we do not discuss here its possible applications.

2 Background

One common task in digital humanities is the creation of digital representations of cultural artifacts, often in the form of transcriptions of existing documents. When we say, though, that a particular resource is a transcription of a particular work, or of a particular copy of that work—a common enough remark—what do we mean? What inferences are licensed by that claim? Given a transcription and adequate knowledge of the transcription practice followed by its creators, what do we learn about the exemplar?

For scholarly editors it may seem that asking questions like this is like asking for answers to the obvious, and that any answer is likely to be trivial. If so, they may be right. We hope, however, that an explicit statement of the obvious may nevertheless be useful in shedding light on basic questions.

Before we proceed, it may be useful to observe that in other fields, the term *transcription* is used with sometimes widely different meanings. In linguistics, for example, it may denote ‘the conversion of spoken words into written language. Also the conversion of handwriting, or a photograph of text into pure text . . .’ In genetics, transcription may mean 'the process of copying DNA to RNA by an enzyme called RNA polymerase (RNAP)', and in music 'either notating an unnotated piece, common in ethnomusicology, or rewriting a piece, either simply recopying (as for clarity), or as an arrangement for another instrument'. Whether a formalization focused on purely textual transcription will illuminate these other forms of transcription remains to be seen. (All quotations are from Wikipedia as of June 2007.)

In the context of scholarly editing, Vander Meulen and Tanselle (1999) offer the following definition:

> ... by *transcription* we mean the effort to report—insofar as typography allows—precisely what the textual inscription of a manuscript consists of. (p. 201)

If understood literally, this definition might seem to take *transcription* to denote an *act* (the '*effort to report*'). However, the term is just as frequently used, also by Vander Meulen and Tanselle in the article just quoted, to denote the *product* of that act, that is, a *document*. Elsewhere, the term is not infrequently used to refer to the *relationship* between documents—one document may be said to be a *transcription of* another document.

It is not unusual for words to refer ambiguously in this way (consider for example words like *drawing*, *painting*, *composition*, or *declaration*), but in this case it means that before attempting to give a more precise account of the meaning of the term *transcription*, it is helpful to decide which of the three one is aiming at: the act, its product, or the relation.

Our aim is to give an account of the relation between an exemplar and its transcription. A formal account of the *act* or *process* of transcription would be at least as difficult as providing a formal account of the process of reading, i.e. possibly impossible. It is clearly a process governed by rules in the sense that it can be done correctly or incorrectly. Some of the obvious rules could probably be formalized: reading the symbols of the exemplar in order (top to bottom, left to right for Latin script). In practice, however, these turn out not to be invariant rules, but heuristics which apply only when applicable. One handbook of transcription, for example [described in Huitfeldt, (1994)], states that:

> Normally, the transcriber
>
> - does not read the graphs of a manuscript one by one
> - does not rely exclusively on the visual evidence in his identification of each letter
> - takes the context into consideration, and
> - draws on his skills and all sorts of background knowledge in numerous, largely unconscious, ways

The relation between exemplar and transcription seems more likely than the reading process to be amenable to formalization.

3 Modeling Transcription

In general, one document (the transcription, *T*) is said to be a transcription of another document (the exemplar, *E*) if *T* was copied out from *E* with the intent, successfully achieved, of providing a faithful representation of a text as witnessed in *E*. Often the purpose is to make some representation which is easier to use than *E* is. For example, *T* may be easier for modern eyes to read, or easier to duplicate. Or *T* may be able to travel while *E* cannot.

The description just given focuses almost entirely on the intent with which *T* is made and the uses to which it may be put; it is silent on just how it is possible for *T* to serve as a record of information derived from *E*. Some sort of similarity must exist between exemplar and transcript, which allows us to draw some inferences about the exemplar by examining the transcript. What is the formal nature of that similarity, and which inferences does it license?

In the simplest cases, *E* has an unproblematic, clearly legible inscription, *T* likewise. The required relation between them is simple: the two documents must contain the same sequence of letters, spaces, punctuation marks, and other symbols.

But documents, as we use the term, are physical objects like manuscripts, typescripts, carved stones, magnetic tapes, disk drives, CD-ROMs, or some portion of such an object. Graphemes, in contrast, i.e. letters, spaces, punctuation marks, and symbols, are abstractions. Clearly, the one cannot 'contain' the other in any but a metaphoric sense. A less metaphoric expression of this basic idea must be slightly more careful and in consequence slightly more cumbersome.

And what does it mean to say that a manuscript symbol is 'the same' as a typed or printed symbol in a transcription of the manuscript? In simple cases, the question rarely arises, yet in transcriptions of historical documents it may be urgent. It may be difficult to decide which letters (if any) are indicated by the various marks on a manuscript page; it may be difficult to decide whether a given mark indicates a letter at all. (Is it just a flourish? A mark made by a slip of the pen? A fly speck?) It may also be difficult to decide what constitutes a letter or symbol. Are the letters *i* and *j* distinct from each other? Short *s* and long *s*?

In order to establish a more precise account of the 'sameness' of sequences of letters, we begin by distinguishing the concepts document, mark, token, and type.

By a document we understand an individual object containing marks. A mark is a perceptible feature of a document (normally something visible, e.g. a line in ink). Marks may be identified as tokens in so far as they are instances of types, and collections of marks may be identified as sequences of tokens in so far as they are instances of sequences of types. In other words, a mark is a token if, but only if, it is understood as instantiating a type.[1]

The distinction among marks, tokens, and types may be applied at various levels: letters, words, sentences, and texts. The identification and separation of the marks, tokens, and types of a document requires a competent reader. A theory of interpretation or of reading is not the concern of our article, however—our goal is only to elucidate what it means to say that *T* is a transcription of *E*, however, the interpretive processes resulting in *T* are properly understood.

We approach answers to those questions by formulating a series of formal models of transcription, which come successively closer to providing a satisfactory account of transcription. The first model focuses on the simplest cases of transcription, ignoring as many complications as possible. Later, those complications are addressed, one by one.

To make the models relatively explicit, we will describe them both in prose and in the formalism of Alloy, a modeling language designed by Daniel Jackson (2006).[2] For the benefit of readers unfamiliar with Alloy notation, English paraphrases of the Alloy declarations will also be provided.

4 The Grapheme-sequence Model

Based on the outline given above of the relation between *E* and *T* and the distinction among documents, marks, tokens, and types, we can formulate a first approximation to an account of the meaning of the statement that *T* is a transcription of *E*, as follows:

(1) *E* and *T* are documents. (It follows that *E* and *T* each contain marks.)
(2) The marks of each document are interpreted as instantiating a sequence of types such that:
   (a) Each mark is identified as constituting one and only one token.
   (b) Each token is identified as instantiating one and only one specific type.
   (c) To the tokens of each document, a total ordering is assigned.
(3) The two sequences of tokens thus identified instantiate the same sequence of types.

The essential proposition here is that a transcription has the same sequence of graphemes (types) as its exemplar. Accordingly, we shall call this first model the 'grapheme sequence' model.

4.1 Basic model

In the following, we elaborate the account and make it more precise by formalizing the model:

(1) *E and T are documents.*
This statement entails first of all that we distinguish some class of things called documents. Formalizing the model in Alloy, we can identify a class with the atoms in a *signature*, and we declare a signature for

documents, with their properties and some constraints on the set, by writing:

```
< 1 Declare Document signature > ≡
  sig Document {
     {Properties of Documents 5}
  } {
     {Constraints on Documents 8}
  }
This code is used in < [File gs.als] 9 >
```

A note on notation may be in order here. The Alloy code is presented here using a 'literate programming' system (Sperberg-McQueen, 1993); code fragments are given numbers and descriptions, and may refer to other fragments. Readers wishing a fuller guide to the notation may consult any of the introductions to literate programming, which may be found on the web, or Knuth (1984).

(2) *The marks of each document are interpreted as instantiating a sequence of types.*
This entails the existence of types. For purposes of this model, the properties of types are unimportant, and we say nothing about them. It is important only that types be distinguishable from each other.

```
< 2 Declare Type signature > ≡
  sig Type { }
This code is used in < [File gs.als] 9 >
```

The model is agnostic about whether the types (and tokens) it is concerned with are those at the character level or those at the level of words and lexical items. This accords, we think, with existing usage of the term *transcription*: some transcribers routinely normalize or regularize the spelling of words in the exemplar, either silently or with explicit marking of such editorial interventions.

Others prefer to restrict the term *transcription* to representations at the graphemic or graphetic level, where such normalization or regularization is not an option. Whether orthographic or other normalization is desirable or not, however, is best framed as a dispute over how most effectively to achieve the purposes of a transcription, rather than as a dispute over the meaning of the term.

(a) *Each mark is identified as constituting one and only one token.*
We could pause here to postulate the existence of marks, and to clarify their relation to tokens, but since it is assumed here that each mark (at least each mark we care about) is a token, and since there are no tokens other than those constituted by marks, the grapheme-sequence model conflates the ideas of marks and tokens. So we will not define a signature for marks.

We declare a Token signature, to represent the class of tokens.

```
< 3 Declare Token signature > ≡
  sig Token {
     {Properties of Tokens 4}
  } {
     {Constraints on Tokens 7}
  }
This code is used in < [File gs.als] 9 >
```

(b) *Each token is identified as instantiating one and only one specific type.*
This is most conveniently done in the formalization with a property defined for Token. Each instance of Token has a *type* property, which specifies the Type to which the Token belongs.

```
< 4 Properties of Tokens > ≡
   type: Type
This code is used in < Declare Token
   signature 3 >
```

Each member of the Token signature can be regarded as having a *type* property, and for a given token *T*, the property can be referred to as '`T.type`', using a dot operator similar to that found in many programming languages for navigating structures. From a formal point of view, *type* can also be regarded as a named relation on the sets Token and Type, and referred to independently of any individual token.

(c) *To the tokens of each document, a total ordering is assigned.*
This statement specifies the relation of documents and tokens: each document is associated with a sequence of tokens. We will give this sequence the name *tokenseq*.

< 5 Properties of Documents > ≡
```
tokenseq: seq Token
```
This code is used in < Declare Document signature 1 >

Just as for the *type* property of Token, so also the *tokenseq* property of Document can be regarded both as a property of an individual document and also as an independent relation. Since sequences are functions from natural numbers to the elements of the sequence, the relation *tokenseq* is a ternary relation on Document, integer, and Token. This will become important later.

(3) *The two sequences of tokens thus identified instantiate the same sequence of types.*
This statement specifies the similarity required to hold between exemplar and transcript. We model this idea by defining a predicate on pairs of documents called *t_similarity*, which holds when the documents have the same sequence of types:

< 6 Define t_similarity > ≡
```
pred t_similar(
  e, t: Document
) {
  e.tokenseq.type
     = t.tokenseq.type
}
```
This code is used in < [File gs.als] 9 >

This predicate may require some explication. The keyword `pred` introduces the declaration of a predicate. The string '`t_similar(e, t: Document)`' gives the predicate's name and identifies its arguments: *e* and *t*, both members of the Document signature. Finally, the definition of the predicate follows, enclosed in braces. The expression '`e.tokenseq`' denotes (as explained above) the sequence of tokens associated with document *e*; the longer expression '`e.tokenseq.type`' composes the *tokenseq* relation with the *type* relation on tokens and types; evaluating the expression yields a sequence of types—specifically, the types mapped to by the tokens of *e*, in sequence. The expression '`t.tokenseq.type`' similarly denotes the sequence of types associated with *t*.[3] The equality symbol has the usual meaning, so the effect of the declaration all in all is to say that two documents are t_similar if and only if they are associated (via *tokenseq* and *type*) with the same sequence of types.

4.2 Preliminary assessment

The definitions given so far describe a number of familiar situations.

For example, consider a document which contains, in order, the token sequence long *s*, *e*, *e*, and short *s*. A transcription which treats long and short s as allographs of the same grapheme (or in other words, which does not distinguish long and short *s*) will map the first and fourth tokens to the type *s*, and the second and third to the type *e*, as shown in Fig. 1.

This phenomenon can be described using the descriptive terminology of the grapheme-sequence model.

On the other hand, the model also has a number of instances with properties which can at best be described as surprising; for example, the instance shown in Fig. 2.

Here, document 1 is as before (Fig. 1), but the universe of discourse also contains two other

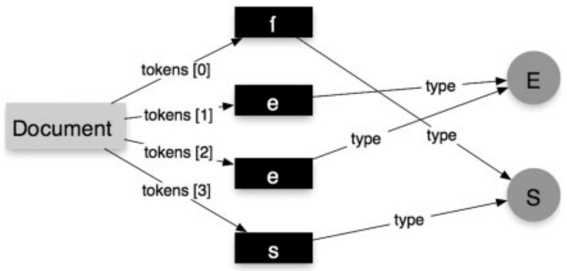

Fig. 1 A simple example of allographs being collapsed

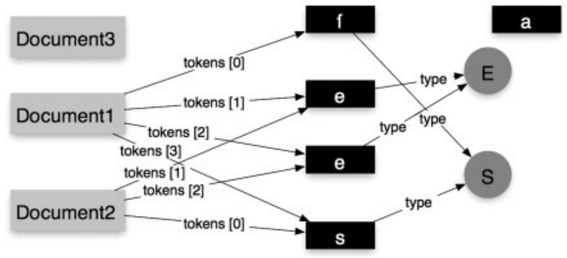

Fig. 2 Another instance of the grapheme-sequence model

documents. Document 2 has the token sequence *s*, *e*, *e*, which would be unremarkable except that the tokens are the same physical marks as those of document 1, in a different order. Our simplifying assumption that documents are physically distinct from each other has consequences for tokens which are not captured in the model. Specifically, no token can occur in more than one document.

Fig. 2 also shows a free-floating token (*a*) which is not in any document. It is not clear this would cause any logical contradictions, but it is hard to understand how a token can exist without existing in some document.[4]

Document 3, in contrast, has no tokens at all. Again, this does not lead to any logical contradictions, as far as we know, but it is hard to believe an object can meaningfully be regarded as a document in our sense if it contains no tokens that can be regarded as instantiating types. The model should reflect this by requiring that documents contain at least one token.

4.3 Refinements to the model

If tokens can occur in at most one document, and must occur in at least one document, then each token must occur in just one document. We can capture this constraint by adding a clause to the definition of Token specifying that for every token *T*, there exists exactly one document *D* with the property that *T* is among the tokens in the token sequence of *D*.

< 7 Constraints on Tokens > ≡
```
    one doc: Document | this
        in elems[doc.tokenseq]
```
This code is used in < Declare Token signature 3 >

The constraint uses the built-in function *elems*, which Alloy provides as a way to map from a sequence of items to the set of items in the sequence.

To exclude empty documents, we wish to specify that the token sequence of any document must be nonempty. It should also be specified that no token appears more than once in the token sequence of its document.[5]

< 8 Constraints on Documents > ≡
```
    #tokenseq > 0
    not(hasDups [tokenseq])
```
This code is used in < Declare Document signature 1 >

The Alloy expressions here say (paraphrased literally) that the cardinality of *tokenseq* (i.e. the length of the sequence) must be greater than zero and that *hasDups* (a function which determines whether a sequence contains any duplicate items) is not true for *tokenseq*.

The entire grapheme-sequence model can be gathered together in a single file, which we will call *gs.als* ('gs' for grapheme sequence, and '.als' is the conventional file extension for Alloy model documents):

< 9 [File gs.als] > ≡
```
    module gs
    {Declare Type signature 2}
    {Declare Token signature 3}
    {Declare Document signature 1}
    {Define t_similarity 6}
    {Assertions about the grapheme-sequence
      model 10}
```

As a sort of sanity check, we can assert some properties of the model. In particular, as we have defined it t_similarity is both symmetric and transitive.

< 10 Assertions about the grapheme-sequence model > ≡
```
    assert t_s_symmetric {
      all d1, d2: Document |
        t_similar [d1, d2] iff
          t_similar[d2, d1]
    }
    assert t_s_transitive {
        all disj d1, d2, d3
            : Document |
          t_similar[d1, d2] and
            t_similar[d2,d3]
          implies
            t_similar[d1,d3]
      }
```

This code is used in < [File gs.als] 9 >

That is, if a transcript *T* is t_similar to its exemplar *E*, then *E* is also t_similar to *T*. In special cases when we are unaware of the dates of origin of two documents which resemble each other, we may not know which is exemplar and which is copy, even if we are convinced that one is a copy of the other.[6]

The transitivity of the similarity relation is also suggestive: if we make a (perfect) transcript *T1* of an exemplar *E*, and then make a (perfect) transcript *T2* by copying *T1*, then *T2* is as similar to *E* as *T1* is.

4.4 Evaluation

So far, this more formal account seems to agree well enough with common usage. In the passage already quoted above, Vander Meulen and Tanselle, for example, describe transcription as 'the effort to report — insofar as typography allows — precisely what the textual inscription of a manuscript consists of' (Vander Meulen and Tanselle, 1999). We can take their phrase 'textual inscription' to denote the physical tokens of the document; the method of reporting the textual inscription of the exemplar is (we propose) to create another document whose textual inscription produces (when rightly read) the same sequence of abstract types.

Even very straightforward forms of transcription require further elaboration of this model (see below), but this very simple model does serve to illustrate some essential features of transcription.

- In a transcription, one letter type is the same as another. Graphetic detail is lost; the only salient information about the token in the exemplar is which type it maps to.

  Some accounts of transcription, and some transcription practices, appear to contradict this claim. As both Driscoll and Haugen describe, for example, some transcription schemes distinguish variant letter forms (long and short *s*, for example, or straight and slanted lowercase *d*) even though they appear to have no basis in phonemic distinctions and are not reliably distinguished in the writing system (Driscoll, 2006; Haugen, 2004). This seems to contradict our claim that all instances of a type are 'the same': when two forms of lowercase *d* are distinguished, then surely one is not the same as the other. It is quite true that if straight *d* and slanted *d* are distinguished, then they are not treated as the same by the transcription system. In terms of the grapheme-sequence model, for such a transcription scheme the two forms of *d* are treated as distinct types. Within each type, the rule continues to apply: one upright lowercase *d* and another *are* treated as interchangeable and indistinguishable.
- The set of letter types distinguished in a transcription is an important defining feature of that transcription, and the choice of an appropriate inventory is one of the basic responsibilities of a transcriber. That choice is a fertile source of disagreements among transcribers interested in similar material. See also the discussion of ligatures in Renear (2001).
- The distinctions usefully made depend on the purposes to which the transcription may be put. The specific forms used for certain letters, for example, may help inform speculation on the origin or exemplar of manuscripts; they provide no information, however, likely to be useful for metrical study or lexicography.
- Transcription necessarily involves the application of competent intelligence, and the selective suppression and alteration of information.

  This is not always recognized. Vander Meulen and Tanselle (1999), for example, regret that:

  > numerous discussions of the subject, including some of the most influential, allow for making certain classes of alteration when transcribing manuscript texts, as if a conscious program of alteration is compatible with the concept of transcription (pp. 201–2).

They overlook, perhaps because it is so obvious, the fact that far from being incompatible with alteration, the concept of transcription largely *consists* in a systematic program of selective alteration, coupled with selective preservation of information. The tokens of the transcript are not those of the exemplar; their physical characteristics are often altered systematically for greater legibility, as when a difficult manuscript is transcribed using a typewriter or in electronic form. The preparation of a transcription, like any representation, entails not just the loss of information, but the addition (or substitution) of new information (e.g. the physical details of the transcript) irrelevant to the exemplar.

- The similarity relation between exemplar and transcript is symmetric and transitive. If we take a sufficiently abstract view of things, we will be unable to decide on purely formal grounds whether some document D1 is a transcript of another document D2, or vice versa. And if D2 is a transcript of D1, and D3 of D2, then for some purposes at least D3 can function as a transcript of D1. Editors who send a transcription to the publisher, rather than an original manuscript, exploit this fact. Scholars who prepare transcripts of manuscripts by working not from the manuscript but from a microfilm or from a digital facsimile exploit an analogous property of facsimiles.
- The abstract text can be distinguished from the concrete document in exactly the same way that the abstract type can be distinguished from the concrete token.
- Because transcription involves the recognition of unitary tokens and their mapping into equivalence classes (types), it has a fundamentally digital character (even when performed with paper and pen); from the perceived continuum of the original, transcription extracts discrete units of information in the form of tokens and types. This property distinguishes transcription from the preparation of facsimiles, which is fundamentally an analog undertaking (although also selective about the properties to be reproduced).[7]
- Some differences in transcription practice may be understood as differences in the choice of the type inventory to be used for the transcription. Some transcriptions operate at the word (lexical item) level, others at the level of characters or character forms.

  The model does not shed much light on the issues surrounding the choice of type inventory. But it does have the virtue of working for both the word and the character level.

On the other hand, the grapheme-sequence model leaves some aspects of transcription unilluminated.

- When a transcription expands an abbreviation, the exemplar and the transcript would seem to have different type sequences.

  In some cases, one might argue that the abbreviation and its expansion are merely allographic variants of each other. Perhaps the graphic form 7 (the Tironian note for *et*, used also in vernacular manuscripts for *and*, *oc*, *und*, etc.), as it appears in a manuscript, and the form *oc* to which a diplomatic transcription might expand it (see, for example, Codex Regius Project, 2006) should be regarded as graphic variants for the same grapheme. It certainly seems plausible to assign a distinct type to the Tironian symbol, but it seems a bit strained to regard the sequence *oc* as an allographic form of that type.

  Alternatively, one could suggest that the transcription of 7 by *oc* is operating on the lexical level, not the character level. But abbreviations are routinely expanded even in transcriptions, which otherwise respect the spelling peculiarities of their exemplars. Ought we to describe such transcriptions as shifting, from one word to the next, from the character to the word level and back? The brackets or italics used in conservative transcriptions to signal the expansion of abbreviations might suggest such a characterization.

  But some abbreviations resist this analysis. When a medieval manuscript has a horizontal stroke or a hooked line over a vowel, and the transcriber supplies the appropriate *n* or *m* or

[*e*]*r*, the level of the transcription has not shifted from character level to word level.

When abbreviations consist simply in the omission of some characters, it is normally the supplied letters only which are italicized or bracketed, not the entire word; this also suggests some analysis other than a shift to the word level.

- Even otherwise conservative transcriptions of historical or literary documents sometimes silently correct obvious or uninteresting typographic errors in their exemplars. Mary Jo Kline (1998) describes two examples:

  Faced with file drawers full of . . . uncorrected copies of outgoing correspondence, the editors of the *Woodrow Wilson Papers* compared surviving carbon and ribbon copies of the same letters and found that Wilson himself had corrected the originals. They concluded that the ribbon copies of these letters had been corrected before dispatch, and the texts were emended accordingly. The *Eisenhower* edition takes a similar approach . . . Mere typographical errors and errors of spelling, which Eisenhower routinely corrected, are emended silently, but editorial corrections of misspellings of proper nouns are accompanied by a footnote recording this emendation.

  This practice reflects the historical editor's particular emphasis on the evidentiary value of documents, as well as the historian's sense that the evidence of the document is better represented by the corrected text (which was presumably what the addressee actually received) than the uncorrected text of the exemplar. But it wreaks havoc with the expectations reflected in the grapheme-sequence model of transcription. In the simple case, the presence of a token of a particular type in the transcript licenses the inference that there is a token of the same type in the exemplar, at a corresponding position in the token sequence. When typographic errors have been corrected silently, that inference is usually true, but not infallibly. It is not clear how best to model the silent correction of errors.
- In practice, transcribers may disagree about how to read a particular token in the exemplar, or even about what marks are to be read as tokens rather than as nonsignificant marks. Such disagreements are hard to explicate by means of the grapheme-sequence model, since it treats the type of a given token as intrinsic to the token, and the sequence of tokens in a document as intrinsic to the document. There would seem to be no way, in this model, for two transcribers to assign the same token to different types: the grapheme-sequence model would appear to decree that such disagreements are not possible: the two transcribers must be talking about different tokens. Similarly, transcribers who disagree about what tokens are present in a document must, on the grapheme-sequence analysis, actually be talking about different documents. These do not seem to be helpful analyses of the disagreements.

  Similar observations apply to disagreements about the correct reading order for a page.

5 The Readings Model

A more elaborate model can be constructed, which addresses some though not all of the shortcomings of the grapheme-sequence model. In this model, *T* is a transcription of *E* if and only if:

(1) *E* and *T* are documents. (It follows that *E* and *T* each contain marks.)

(2) A *reading* of a document interprets the marks in that document as instantiating a sequence of types:

  (a) The reading identifies some marks as constituting tokens.

  (b) The reading assigns a total order to the tokens.

  (c) With each token it identifies in the document, the reading associates a type.

  (d) The sequence of tokens, taken together with the token–type mapping, generates a sequence of types.

(3) Some reading of *E* and some reading of *T* exist, which generate the same sequence of types.

5.1 Formalization of the model

Again, we elaborate the model while formulating it in Alloy notation.

(1) *E and T are documents.*

In this model, documents have no internal structure of interest to the model.[8]

The declaration of the signature is very simple.

< 11 Declare Document signature > ≡

```
sig Document{}
```

This code is used in < [File readings.als] 26 >

(2) *A reading of a document interprets the marks in that document as instantiating a sequence of types.*

We must assert the existence of readings. They will have some properties and constraints which will be outlined below.

< 12 Declare Reading signature > ≡

```
sig Reading {
    {Properties of Readings 14}
  } {
    {Constraints on  Readings 16}
  }
```

This code is used in < [File readings.als] 26 >

(a) *The reading identifies some marks as constituting tokens.*

We assert the existence of tokens, and stipulate (as a simplification of a more complex reality) that each token is in exactly one document.

< 13 Declare Token signature > ≡

```
sig Token {
  {Mapping from Token to Type 23}
   d: one Document
} {
  {Constraints on Tokens 17}
}
```

This code is used in < [File readings.als] 26 >

The reading is similarly a reading of exactly one document.

< 14 Properties of Readings > ≡

```
doc: Document,
```

Continued in <Token sequence as property of a reading 15>, <Token-type mapping as property of Reading 18>, <Type sequence as property of Reading 21>

This code is used in < Declare Reading signature 12 >

We will need to ensure that the tokens sequenced by a reading are in the document associated with that reading. Otherwise, some instances will have inconsistent properties.

(b) *The reading assigns a total order to the tokens.*

As before, we call the sequence of tokens *tokenseq*.

< 15 Token sequence as property of a reading [continues 14 Properties of Readings] > ≡

```
tokenseq: seq Token,
```

And as before, we require that it be non-empty and free of duplication:

< 16 Constraints on Readings > ≡

```
#tokenseq > 0
not(hasDups [tokenseq])
```

Continued in <Constraint on tt and tokenseq 19>, <Constraints on token-type mapping 22>

This code is used in < Declare Reading signature 12 >

We should also specify that a token is associated with a particular document if and only if it appears in the sequence of tokens identified by some reading of that document. This ensures that the information in the *d* property of tokens is consistent with the information provided by the *doc* and *tokenseq* properties of readings.[9]

< 17 Constraints on Tokens > ≡

```
  all r: Reading |
    this in elems
      [r.tokenseq]
    iff r.doc = d
```

Continued in <Further constraint on Tokens 24>

This code is used in < Declare Token signature 13 >

(c) *With each token it identifies in the manuscript, the reading associates a type.* We will call this token–type mapping *tt*.

< 18 Token-type mapping as property of Reading [continues 14 Properties of Readings] > ≡

```
tt: Token -> Type,
```

The relation *tt* must provide a type mapping for every token in the token sequence identified by the reading (*tokenseq*), and it has no occasion to provide type mappings for any other tokens. So the domain of *tt* ('`dom[tt]`') should be identical to the range of *tokenseq*, that is, to the set of elements in the sequence ('`elems [tokenseq]`'). Moreover, it should be a function on those elements: each should map to exactly one type.[10]

< 19 Constraint on tt and tokenseq [continues 16 Constraints on Readings] > ≡

```
dom[tt] = elems [tokenseq]
function[tt, elems
[tokenseq]]
```

We have referred to types in the declaration of tokens, so we must also assert the existence of types.

< 20 Declare Type signature > ≡

```
sig Type {}
```

This code is used in < [File readings.als] 26 >

(d) *The sequence of tokens, taken together with the token–type mapping, generates a sequence of types.* First, we declare *typeseq* (a sequence of types) as a property of readings:

< 21 Type sequence as property of Reading [continues 14 Properties of Readings] > ≡

```
typeseq: seq Type
```

Then, we define its required content; this appears among the constraints on readings:

< 22 Constraints on token-type mapping [continues 16 Constraints on Readings] > ≡

```
typeseq = tokenseq.tt
```

The sequence of types is generated by composing the sequence of tokens *tokenseq*, viewed as a relation on integers and tokens, with the token–type mapping *tt*.

Since each reading determines a distinct token–type mapping, the actual relation described by *tt* is a triple of reading, token, and type. It is instructive to view this same triple from the point of the view of the token, in which it appears as a mapping from readings to types. (When considering a particular document, one may say of a given token that editor X reads it as a long *s* and editor Y as an *f*.) This is conveniently expressed as a property of the token by the property *rtt* ('reading–token–type mapping'):

< 23 Mapping from Token to Type > ≡

```
rtt: Reading -> Type,
```

This code is used in < Declare Token signature 13 >

To avoid confusion, we wish to ensure that the reading–token–type triples in the *rtt* property of tokens are the same (except for the order of the columns within the triples) as in the *tt* property of readings. We can do this with a constraint on tokens:

< 24 Further constraint on Tokens [continues 17 Constraints on Tokens] > ≡

```
all r: Reading |
  all t: Type |
    r -> t in rtt
    iff
    this -> t in r.tt
```

(3) *There is some reading of E and some reading of T which generate the same sequence of types.*

This rule is captured in the definition of t_similarity.

< 25 Define t_similarity > ≡

```
pred t_similar(
  e, t: Document
) {
  some r1, r2: Reading | {
    r1.doc = e
    r2.doc = t
    r1.typeseq
      = r2.typeseq
    }
  }
```

This code is used in < [File readings.als] 26 >

The overall structure of the readings model is shown in this top-level fragment:

< 26 [File readings.als] > ≡

```
module readings
open util/relation
  as relation
```

{*Declare Type signature 20*}
{*Declare Token signature 13*}
{*Declare Document signature 11*}
{*Declare Reading signature 12*}
{*Define t_similarity 25*}

Here, the line '`open util/relation as relation`' causes an Alloy library file to be included in the definition of the model; the *util/relation* file defines several functions on relations, e.g. for range (*ran*) and domain (*dom*). We have appealed to some of these functions above.

5.2 Evaluation

The readings model illustrates all the aspects of transcription illustrated by the grapheme-sequence model. Additionally, some further aspects of variation in transcription practice can be described in the terms it offers.

Transcribers may disagree about whether a particular mark in the document is or is not to be read as a letter (as opposed to an accidental mark of the pen, or a flourish, or some other discoloration); in the model, such disagreements are visible as differences in the set of tokens attributed to a document by the transcribers' different readings of the document.

Transcribers may agree that a particular mark in the document is a token, but may disagree about what it is. The readings model describes such disagreements as differences in the token–type mapping of two readings.

Transcribers may agree on the set of tokens to be read, and about the types they are to be attributed to, but disagree about the order in which the tokens are to be read. For example, consider a manuscript with the lines

he has a
  whitewashed
small house

One transcriber may follow the usual top-to-bottom, left-to-right rule of most writing in the Latin alphabet and read this as 'he has a whitewashed small house'. Another may note the fact that 'whitewashed' is not quite flush to the left margin and is centered (more or less) over the space between 'small' and 'house', and these, together with the slight awkwardness of the phrase 'a whitewashed small house' may lead the second transcriber to read the manuscript as saying 'he has a small whitewashed house' (despite the fact that there is no caret or other insertion mark in the exemplar). Such decisions depend on a project's transcription policy, on knowledge of the author's or scribe's usual habits, and sometimes on one's understanding of the text. (If the surrounding text expounds the issue of adjective order in English, it may make clear which of these transcriptions is correct.) Most of these influences are (rightly) invisible to a formal treatment of transcription: they mark the difference between a correct transcription and an incorrect one, a more useful and a less useful transcript, not the difference between a transcription and a block of wood. But the formal model of transcription should allow us to say what it is that the transcribers are disagreeing about, or where their disagreement appears in the formal model (here, in arranging the same tokens in a different sequence in the two readings).

On the other hand, several gaps remain in the readings model. For example, it still does not model differences of practice in the expansion of abbreviations. And it provides no obvious solution to some other problems, which are discussed in the following section.

6 Projections

Some transcriptions systematically omit certain kinds of material from the exemplar. In these transcriptions, drawings, mathematical formulas, encrypted material, and the like may be omitted with a terse notation (e.g. '[FIGURE]' or '[EQUATION]') or passed over in silence. How should such omissions be modeled?

And some transcriptions provide explicit markings for the line- and page-breaks of the original, so that if the example given above began on the 20th line of the page, the first transcription might be given as '$|^{20}$ he has a $|^{21}$ whitewashed $|^{22}$ small house'.

Paleographic transcriptions often include vertical bars to signify line breaks in the original; the sequence of types in the transcript thus appears to include items not present in the type sequence of the original. The line breaks of the original could conceivably be interpreted as a particular kind of letter token; this would restore the equality of the two type sequences, and would illustrate one way to deal with layout and other 'suprasegmental' properties of the exemplar. But transcriptions which mark line breaks often also number them. And while treating a line break in the exemplar as a particular form of symbol is plausible, if a bit strained, treating it as a symbol not just for line break but for 'line 1', 'line 2', etc. seems unhelpful. It will be more useful to say that the graphemic inventory of the transcript is partitioned into one set of graphemes, which correspond to tokens in the original, and one which do not. It is the sequence of types of the first kind which must be similar to that of the exemplar.

What should a formal model say about such additional metadata?

One might attempt to address these problems by further elaborating the model and partitioning the tokens of each document (or the types of each type inventory) into normal characters and metacharacters. A document might then exhibit two different token or type sequences of interest: the full sequence, and the one which omits all metacharacters. A projection function could be described, which when given the full sequence would produce the shorter sequence of 'normal' characters. The similarity relation holding between E and T would be identity of the 'normalized' type sequence.

We have worked out parts of such an elaboration, but we do not include it here, because it has proven an unnecessary complication.

Readings can already (in the readings model) differ in identifying different sets of tokens in the original. Perhaps we should free ourselves from the silent assumption that all transcriptions are full transcriptions of everything in the document, and accept instead that some transcriptions are intentionally incomplete. Then the difference between a transcription which includes mathematical equations and one which omits them is simply that the sets of tokens each chooses to read in the exemplar differ systematically.

The same principle of selective reading can be applied in the case of diplomatic transcriptions which mark line breaks: when checking to make sure that the transcript and the exemplar have (readings with) the same sequence of types, we skip over the explicit marking of line breaks in the transcript; we do not *read* them, because the vertical bar characters and the immediately following superscript numbers 'do not count'.

7 Conclusions and Future Work

The models given here capture some of the fundamental formal properties common to all transcriptions, and they seem to us to provide a potentially useful basis for further work.

A more fully satisfactory formal account of transcription must deal with a number of complications not addressed in the grapheme-sequence and readings models. Several issues stand out as in need of attention.

As already noted above, the expansion of scribal abbreviations poses a challenge for our formal treatment of transcription, because it destroys the one-to-one correspondence between types read in the original and in the transcript. One possible elaboration would be to segment both the type of the exemplar and that of the transcription, and insist on a pairwise equivalence of the segments, where two segments are equivalent if identical or if one is an abbreviation for the other. Abbreviation forms like nasal strokes, which are properly expanded in different ways in different contexts, pose a problem for that approach, but perhaps not an insuperable one.

Transcriptions of damaged materials may distinguish fully preserved characters from partially preserved characters, with or without surmises as to the identity of the character. Transcriptions of some writing systems must account both for systematic ambiguity of tokens in the writing system and for uncertainty as to which token is actually present. These cases can, it is believed, perhaps be modeled as instances of writing systems which present special difficulties for the choice of the appropriate type inventory. Further work is needed to check that this is so.

When compared with the reality of transcription, the treatment of the type and token systems given above seems too simple. In reality, the transcriber who distinguishes long and short *s* and the transcriber who collapses the distinction share far more premises than the readings model suggests: both transcribers will be acutely aware (if they are working with conventional uses of the Latin alphabet) that long and short *s* are allographs. Even the conservative transcriber who preserves the distinction will be aware that they represent, at some more abstract level, the same grapheme. Moreover, a transcription which collapses long and short *s* can be created automatically from one which preserves the distinction, without re-consultation of the exemplar. Often, when a transcription is intended for more than one use, this fact is exploited.

The readings model suggests an account of this situation in which the two readings of the exemplar use different inventories of types and thus different token–type mappings. In this formal view, the close relation of the two inventories (one preserves all the information present in the other, and adds a further distinction) appears as an accident, instead of as an essential property of the systems of types used. In practice, transcription policies are often carefully designed with multiple related sets of types, and thus multiple token–type mappings, in mind. An ideal formal model of transcription would have more to say about this topic.

A more fundamental problem is given by the silent assumption, in the preceding remarks, that the text represented in a particular document, and thus the document itself, can usefully be represented as a one-dimensional sequence of types. This view, while still held by some prominent theorists of hypertext, and others, appears painfully simplistic in the light of recent years' work on markup languages and text encoding. It is not the case that even in a conventional printed book each character except the last has a single character which is 'the next character'. Footnotes, running headers and footers, page numbers, catch words, and other phenomena frequently create situations structurally more complex than simple sequences. One of the key challenges for any formal treatment of transcription will be to make it fit into the more powerful notions of textual structure reflected in modern markup systems; on this topic see also Renear (2001).

Any attempt to address that challenge, however, must cope with the fact that different markup systems may have quite different notions of textual structure; some method of abstracting away from unimportant details seems likely to be needed, in order to make it possible to compare transcriptions prepared in different markup systems. (The alternative would seem to be saying that transcriptions prepared in different markup systems can never agree on anything.) Some differences in the markup will reflect differences of transcription, and possibly disagreements about the correct reading of the exemplar; others will be irrelevant to questions of transcription. We will need a way to tell which is which.

References

Codex Regius Project. (2006). *GKS 2365 4to: Vǫluspá: an electronic edition*, Prepared by the Codex Regius Project. Menota (Medieval Nordic Text Archive) Ms. 4, version 0.9, 21 December 2006. Bergen: Medieval Nordic Text Archive.

Devlin, K. (1991 reprint 1995). *Logic and Information*. Cambridge: CUP.

Driscoll, M. (2006). Levels of transcription. In Unsworth, J., O'Brien O'Keeffe, K., and Burnard, L. (eds), *Electronic Textual Editing*. New York: MLA.

Haugen, O. E. (ed.) (2004). *The Menota Handbook: Guidelines for the Electronic Encoding of Medieval Nordic Primary Sources. Version 1.1*. Bergen: Medieval Nordic Text Archive. http://www.hit.uib.no/menota/guidelines (accessed 20 August 2008).

Huitfeldt, C. (1994). Toward a Machine-Readable Version of Wittgenstein's Nachlass: Some Editorial Problems. In Senger, H. G. (ed.), *Philosophische Editionen. Erwartungen an sie – Wirkungen durch sie.* [Philosophical editions: expectations and effects]. Beihefte zu editio Band 6 [Supplementary volumes to the journal editio, 6]. Tübingen: Max Niemeyer Verlag, pp. 37–43.

Huitfeldt, C. (1995). Multi-dimensional texts in a one dimensional medium. *Computers and the Humanities*, **28**: 235–41.

Huitfeldt, C. (2006). Philosophy case study. In Unsworth, J., O'Brien O'Keeffe, K., and Burnard, L. (eds), *Electronic Textual Editing*. New York: MLA.

Jackson, D. (2006). *Software Abstractions: Logic, Language, and Analysis*. Cambridge: MIT Press.

Kline, M.-J. (1998). *A Guide to Documentary Editing*. Baltimore: Johns Hopkins University Press (Second edition).

Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, **27**: 97–111. Rpt. [rev.] in his *Literate Programming*, CSLI Lecture Notes Number 27. [Stanford, California]: Center for the Study of Language and Information, 1992, pp. 99–136.

Peirce, C. S. (1906). Prolegomena to an apology for pragmaticism. In Hartshorne, C. and Weiss, P. (eds), *Collected Works*. Vol. 4 Cambridge, MA: Harvard University Press, pp. 1931–58.

Renear, A. (2001). Literal transcription—Can the text ontologist help? In Fiormonte, D. and Usher, J. (eds), *New Media and the Humanities: Research and Applications*, Proceedings of the first seminar computers, literature, and philology, Edinburgh, 7–9 September 1998. Oxford: Humanities Computing Unit, University of Oxford, pp. 23–30.

Robinson, P. (1994). *The transcription of primary textual sources using SGML*. Oxford: OHC, Office for Humanities Communication Publications, Number 6.

Robinson, P. and Solopova, E. (1993). Guidelines for the transcription of the manuscripts of the Wife of Bath's Prologue. In Blake, N. and Robinson, P. (eds), *The Canterbury Tales Project Occasional Papers Volume I*. Oxford: OHC, Office for Humanities Communication Publications, Number 5.

Sperberg-McQueen, C. M. (1993). SWEB: an SGML Tag Set for Literate Programming. *Unpublished working paper, 1993, revised 1994–96*. http://www.w3.org/People/cmsmcq/1993/sweb.html (accessed 20 August 2008).

Stevens, M. E. and Burg, S. B. (1997). *Editing Historical Documents: A Handbook of Practice*. Walnut Creek, California: Altamira Press.

Vander Meulen, D. L. and Tanselle., G. T. (1999). A system of manuscript transcription. *Studies in Bibliography*, **52**: 201–12.

Notes

1 The *types* referred to here are not to be confused with the datatypes of programming languages, nor with the types in Russell's theory of types; they do, however, include the 'types' of the type/token distinction introduced by C. S. Peirce, for which the locus classicus appears to be (Peirce, 1906, pp. 423–24).

2 For our purposes the choice of modeling formalism has relatively little importance; we could as easily have written equivalent models using first-order predicate calculus, Z, ACL2, or any of a host of other

formalisms, instead of or in addition to Alloy; we are far from any level of detail or style of specification which would make the differences in expressive power among these notations visible. We use Alloy because it has a simple notation and useful tools.

3 For reasons of space, we cannot give a full account of Alloy notation; the interested reader is directed to Jackson (2006) or to the excellent documentation on the Alloy project web site at http://alloy.mit.edu/ (accessed 20 August 2008).

4 The rule requiring tokens to occur in documents may be uncontroversial.

The rule requiring them to occur in at most one document may, in contrast, seem questionable. One can readily imagine situations in which documents comprising loose pages may be difficult to disentangle from each other; indeed, the situation is not uncommon when the papers of an author have fallen into disarray; the papers of C. S. Peirce are a familiar example. In such situations, different observers might attribute a particular sheet of paper to two different documents. Would that not mean that the tokens on that page occurred in two different documents?

We believe the answer is usually no. When one observer claims the page for one document, and the other for a different document, they normally believe themselves to be disagreeing. If it were normal for a given page (and its tokens) to be part of more than one document, no such belief would be necessary.

On the other hand, electronic documents also frequently take the form of a collection of independent objects (e.g. files, included by some driver file). In the electronic context, there is no logical objection to including the same material into multiple documents; indeed, it may be standard practice for boilerplate text like copyright information to be embedded in many different documents.

Relaxing the constraint given in the text to `all t: Token | some d: Doc | t in d.tokens` would allow this situation to be modeled. But for now, we refrain from this generalization in the interests of simplicity, and leave the extension to shared tokens as an exercise for the reader (or for the future).

5 Like the rule about tokens occurring in at most one document, this rule about tokens occurring at most once in the token sequence may be controversial. Acrostics and related forms appear to violate the rule, since the letters or words appearing in the acrostic must be read twice, once in the normal reading order and once in the hidden word. Strictly speaking, however, acrostics should probably be modeled not by allowing tokens to appear more than once in the sequence, but by allowing documents to contain multiple sequences of tokens, and allowing or requiring the tokens of the acrostic sequence to form a subset of those of the main sequence.

6 Such uncertainty is rare in general, but that is largely because we often have information about the origin of the documents we study. When such information is lacking, establishing the relative priority of documents can become an important scholarly task. Such work often exploits imperfections in the transcription process—imperfections, it may be pointed out, that none of the formalizations described here successfully model. For the models given here, an imperfect transcription is not really a transcription at all.

7 We are aware, of course, of hybrid forms which combine digital and analog properties; for example, some transcriptions seek to reproduce the layout of the page, especially when the reading order of the page is not clear. Here, as in many other cases, the transcriber resorts to analog reproduction when the exemplar is imperfectly understood. Further discussion of the digital/analog distinction and its relation to cognition, and pointers to further reading, may be found in Devlin (1991, pp. 17–19).

8 Documents certainly have important physical properties, and understanding those properties may be crucial to correct transcription. But they seem to affect the process of reading more than the formal characterization of the result. (In any case, it is not at all clear how to model physical properties formally; they do not seem to form a set in the usual sense.)

9 The *d* property of tokens could be omitted as redundant; we retain it for convenience.

10 By requiring that each token map to exactly one type, we leave out of consideration all those cases in reality in which the correct reading of a token is partially or utterly uncertain. Effectively, the readings model says that such transcriptions are not complete, or not transcriptions. This simplifies the formal treatment, but is not wholly satisfactory.