An audio view of (LA)TEX documents.

T. V. Raman
Center for Applied Mathematics
Cornell University
Ithaca NY 14853–6201
Phone: (607)255–7421
Internet: `raman@cs.cornell.edu`

Abstract

Till now, (LA)TEX has been used to generate typeset documents. This paper points out an alternative view — the generation of audio renderings. The (LA)TEX typesetting source captures a lot of document structure which is then used to typeset the document. This can be exploited to convey document structure in audio renderings of the document as well. This becomes especially important when dealing with mathematical documents. Our approach attempts to develop notions of audio formatting similar to the well understood notions of visual formatting and layout. Our goal is to do for the audio document what (LA)TEX has done for the printed document.

Introduction

Lack of ready access to current mathematical and technical literature has been a major stumbling block throughout my career. I first became interested in trying to use (LA)TEX to overcome this problem when I received the LATEX source for the lecture notes in a course on the Design and Analysis of Algorithms at Cornell. I use a talking computer, since I am visually handicapped, and availability of technical documents on-line meant that I could have them read by my computer. However I soon realised the futility of having the computer read out (LA)TEX source directly. The only other effective solution available at the time, namely to have the printed text read aloud, was clearly inadequate. Of course I had the option of trying to get the printed output read out using a reading machine, but in the case of texts with heavy mathematical content, this continues to remain impracticable at present. Current day OCR[1] technology is incapable of handling typeset mathematics.

Using the (LA)TEX sources as a starting point for generating audio renderings was a temptation too hard to resist. (LA)TEX captures a lot of syntactic and sometimes even semantic information about the document content, and this information can be used for more than just typesetting the document. It could be used equally well in producing high quality computer-generated audio renderings of the document. This is especially true when it comes to reading mathematical texts. Complicated mathematical constructs, which prove a major stumbling block for conventional OCR technology, present far less of a problem, since the (LA)TEX constructs that are used to produce the final output capture a lot of information about what is being laid out on paper. Thus this information can be effectively captured at the (LA)TEX source level and exploited in producing audio. I therefore started work on a system for generating audio renderings of technical documents presented in the form of (LA)TEX mark-up source.

[1] Optical Character Recognition

A first attempt

I first worked on a program that would transform (LA)TEX source to a form more suitable to be read out by a talking computer. Reading the (LA)TEX source directly is impractical, since you have to listen to a stream of "backslash" and other extraneous utterances.

This first attempt resulted in the development of TEXTALK [Ram91], a program that I continue to use for my day-to-day work as a graduate student at Cornell. This program carries out simple transformations on the (LA)TEX source and the resulting text can be viewed using standard UNIX tools. The text which is displayed on the screen can then be effectively read out by the talking computer. Thus presented with the expression:

$$\frac{1+\sqrt{5}}{2}$$

the program transforms the above text to:

> *The fraction with numerator 1 + square root of 5 and denominator 2 end of fraction*

This is much more intelligible than the following cryptic utterance which would be generated if the (LA)TEX source were being directly read.

> *dollar dollar backslash frac left brace one plus backslash sqrt left brace five right brace right brace left brace two right brace dollar dollar*

TEXTALK modified (LA)TEX mark-up source corresponding to mathematical notation and generated text that would parallel what a human reading out the printed result would say. See the appendix, "An example of complex math expressions", for a more detailed example. I obtained immediate direct access to several text books whose authors kindly agreed to make available the on-line sources. Currently I also have access to the on-line sources for the AMS bulletins. Thus the program makes the latest mathematical publications accessible to me.

This first attempt uncovered a lot of interesting issues, which then led naturally to the next step, namely, developing notions of audio formatting analogous to the well understood notions of visual formatting. Though the program as it works is eminently usable, the audio renderings generated are not as effective in conveying the complete structure of the document. Sub-expressions occurring in a complicated expression make perfect sense when handled by the system, but it is still difficult to comprehend extremely complicated mathematical expressions, especially in terms of understanding how the various sub-expressions interact with one another. The example in the appendix, which shows the text generated for a complex math expression, makes these shortcomings explicit.

A rigorous approach

The initial implementation revealed the need for developing a rigorous approach to audio formatting. It also became apparent that in order to do full justice to the audio, carrying out transformations based on a linear scan of the (LA)TEX source itself was not adequate. Even though (LA)TEX source contains a lot of useful information about document structure, the earlier approach of string substitution, i.e., scanning the source and applying simple transformations fails to fully exploit all of this information. The transformations carried out tended to be local in nature as string substitution ignores global structure and this was identified as a principal cause of the ineffectiveness in conveying global structure. The above becomes clear when we compare the readings corresponding to a complicated expression generated by a trained and experienced reader with that generated by the system. See the appendix, "Text taken verbatim from the RFB recording" and "Transformed text generated by TEXTALK", for a comparison. In fact even a straightforward transcription of the text as present in the recordings from the RFB[2] loses a lot of information. This is because the trained reader inserts appropriate pauses and uses other prosodic cues to convey the nesting of complicated sub-expressions. This shows clearly that a system that attempts to read out the text resulting from carrying out simple transformations to the (LA)TEX source will be unable to convey such structural information. The trained RFB reader is able to insert appropriate cues into the spoken text only after having "parsed" and "re-parsed" the expression. This shows a clear need for first constructing higher level representations of the expression in order to improve the quality of the audio rendering. Thus there are two steps to audio rendering:

- Generate the text to be spoken.
- "audio format" this text, i.e., insert appropriate cues into the spoken text in order to convey structure.

A two-step approach. The preceding discussion shows that a better approach is to subdivide the problem into:

1. Build a high-level model of the document by parsing the (LA)TEX source.
2. Generate audio renderings by applying appropriate audio formatting rules to the resulting structure.

Advantages. This approach no longer suffers from the drawbacks alluded to earlier. This is because the audio renderings that are now driven off a high level representation of the document can draw upon global knowledge of document structure. Thus while sub-expressions continue to be formatted for audio by applying transformations as before, the audio renderings can now capture information about how various entities appearing in the document relate to one another. The human reader conveys such structural information using prosodic cues. When using computer-generated speech, the range of prosodic cues that we can use is limited in comparison. However these can be augmented by using non-speech audio cues. By *non-speech* audio cues, we mean short chords of music, beeps etc. that can be used to convey non-textual content. These can prove extremely useful, since they can be used to

[2] Recordings for the Blind

cue the listener to extra-textual content without introducing unnecessary verbiage in the audio renderings. In addition, the audio documents generated from such high level document structures will be capable of allowing the user to browse the document.

Generating high-level document models from (LA)TEX source

This section details the approach we have taken to solving the first of the two subproblems outlined in the previous section. Well-written (LA)TEX documents capture document structure using macros designed to reflect logical rather than layout structure. This fact is heavily relied upon when generating high level models given specific instances of (LA)TEX documents. This forces certain constraints on the type of (LA)TEX documents that such a system will be able to handle. These constraints are derived in the following subsections. We then give an approach for generating such high-level models for (LA)TEX documents satisfying these constraints.

Defining high-level models. We think of high-level models of the documents as being given by abstract syntax trees corresponding to a specific language $\mathcal{L}$. Different document types satisfy different languages, and thus have different high level representations. The abstract models we define are thus specific to a given class of documents. For the present we will consider the *article style* of LATEX.

A hierarchical structure. Documents conforming to the *article style* of LATEX have a clear hierarchical structure. The document divides neatly into a simple tree structure where the subtrees correspond to various structural units such as section, subsection, etc. LATEX documents have this structure clearly tagged by the use of standard LATEX constructs and this allows us to get at the high level structure [Lam86]. Similar structures are also easy to obtain from well written TEX documents where the structure is marked up using macros from standard packages such as *plain TEX* [Knu84]. However, since TEX does not always insist on the use of predefined structures for marking up the document, this step can prove difficult when dealing with "raw" TEX documents.

Defining a recognizable class of (LA)TEX documents. TEX as described by Knuth [Knu84, Knu86] is a powerful typesetting language and embodies many features of a programming language. By providing primitives normally found in a programming language it affords immense flexibility to the designer of a document. However, with this flexibility come a lot of problems, since such power in the hands of an average user can prove dangerous.

> All power corrupts and absolute power corrupts absolutely!

This statement is true in the world of TEX documents as well. A properly prepared TEX document uses the power of the language sparingly and avoids mixing typesetting commands with document content. Using well-designed formats results in (LA)TEX source that clearly reflects the document structure. However the ease with which new constructs can be defined in TEX means that the above principles are often violated.

Documents which rely on absolute commands like `\vskip` to achieve document structure by providing the right visual effect present serious problems to a program that is trying to build a higher level model of the document. This is to be expected since the use of such absolute commands within the text of an electronic document indicates an assumption that the electronic source will be used only for typesetting the document. Thus the first and most important constraint that we impose on the class of (LA)TEX documents that we handle is that document structure be clearly tagged using only standard macro packages. Thus our current parser recognizes an enumerated list in LATEX which is clearly marked up, i.e., each new item is explicitly tagged using `\item`. However if an author chooses to mark-up certain items in an enumerated list by using `\item` and other items by `\foo`, where `\foo` has been defined by the author to generate the right visual effects required to signal a new item, the parser fails to recognize some of the items in the list.

Classifying (LA)TEX macros. The level of complexity present in constructing an abstract model of a document given its (LA)TEX source is directly determined by the type of macros used by the author. Macros can be used to mark-up document structure, for laying out specific structures and objects, and for achieving visual effects. As pointed out above, macros are also used to augment predefined formats. In addition macros are also widely used to make the task of key-boarding easier. This multiple use of macros causes some difficulty when constructing high level models of a (LA)TEX document. In order to define the class of (LA)TEX documents we can handle, we need to classify TEX macros according to how they are used. This will allow us to clearly define the class of macros that we can handle in our document recognition step and will in effect define the class of documents that the system will be able to recognize. Further, this classification step will also indicate to what extent we should expand macros during the recognition step and how the abstract model is to capture macro calls.

In the following, $\mathcal{M}_U$ is used to denote the universal set of all macros. The various subsets are denoted by appropriate suffixes.

Graphic. This subset will be denoted by $\mathcal{M}_G$. Macros that provide primitive typesetting operations such as the `\kern` and `\hrule` macros in TeX are typical examples of such macros. They are characterized by their visual nature. Explicit use of such macros in a document does not provide sufficient information to allow for the construction of an abstract model.

Graphic macros representing standard objects. This set of macros will be denoted by $\mathcal{M}_O$.

The term *standard object* is used here to refer to commonly occurring entities such as integrals and fractions. These objects have a standard visual template according to which they are typeset, and a properly composed document renders these visually by using specialized macros that take appropriate arguments. Thus by their very nature such macros capture a functional representation of the object. When confronted by such macros in a document, it is unwise to try and expand them any further in terms of the lower level control sequences from which they have been constructed. Thus the `\frac` macro of LaTeX contains all the higher level information we can get about the object it renders, and the model we build should capture this call.

Another difficult subset of macros that belong to this category are special symbols built up with primitive control sequences. This is typical of complicated combinations of primitive macro calls that are used to create special visual effects. When building the higher level model of the document, we need to capture the essence of what is being conveyed by the use of such macros, rather than the result of expanding the macro call. Thus if `\Real` has been built up using a set of primitive *visual macros* to produce a real number symbol, the abstract model should stop by capturing just the macro call `\Real` rather than attempting to expand it any further. In fact expanding the call will actually eliminate information.

In an ideal world the principal type of macros one would encounter when parsing the typesetting source would be elements of $\mathcal{M}_O$. However life is not so simple, and often electronic documents abound with the use of lower level control sequences. Further $\mathcal{M}_G \cap \mathcal{M}_O = \emptyset$ does not always hold. It can be argued that the `\Real` control sequence discussed earlier actually belongs to $\mathcal{M}_G$. This would be true if we did not know what the author intended to represent by the use of the macro, which could often be the case.

The above is also true of the use of TeX control sequences like `\atop` and `\over` which often require some knowledge of the context in which they are used in order to come up with a semantic interpretation of their use. Thus the use of `\atop` in a nested subscript often means conjunction, while it means something entirely different when used in rendering a Legendre symbol. Consider the following example:[3]

$$\sum_{{\scriptstyle 1\le i\le p \atop \scriptstyle 1\le j\le q} \atop \scriptstyle 1\le k\le r} a_{ij}b_{jk}c_{ki}$$

which is produced by:

```
$$\sum_{{\scriptstyle1\le i\le p
\atop\scriptstyle1\le j\le q}
\atop\scriptstyle1\le k\le r}
a_{ij}b_{jk}c_{ki}$$
```

Macros for simple text substitution. This class of macros will be denoted by $\mathcal{M}_T$. These macros are typically used to make the task of key-boarding easier. In most situations it is safe to expand these macros since their expansion does not lead to loss of structural information. However, once again $\mathcal{M}_O \cap \mathcal{M}_T = \emptyset$ is not always true.

Consider the use of the macro `\reader` in the following sentence.

"We are working on a `\reader` for electronic documents. "

where `\reader` has been defined elsewhere as

```
\def\reader{new exciting reading machine}
```

This macro clearly belongs to $\mathcal{M}_T$ in the context in which it is described. Expanding it directly will lead to the text "We are working on a new exciting reading machine for electronic documents." This can now be easily rendered in audio.

However, viewed from a different perspective, i.e., the use of `\reader` as a logo, this macro could well be said to be in $\mathcal{M}_O$. In the context of audio formatting, we may wish to render the result of the call to the `\reader` macro differently.

Recognizing the type of a macro. Recognizing the class to which a given macro belongs is a hard problem. In general no universal classification

[3] Taken from *The TeXbook*, Chapter 17.

will always hold, as is clear from the previous description. However we can use some simple heuristics to classify a major subset of the commonly occurring macros. Clearly all the primitive typesetting operations provided by TEX in terms of putting marks on paper belong to $\mathcal{M}_G$. Further macros that are known not to be in $\mathcal{M}_G$ but take arguments are typically in $\mathcal{M}_O$, since the use of macro arguments normally indicates that a template is being filled up. Macros that which do not fall into either category now fall into $\mathcal{M}_T$. Given more information about the context in which the macro is being used, it becomes possible to further refine the above classification.

Summary of constraints. To summarize, here are the constraints we need to impose on the class of (LA)TEX documents we can handle:

1. Document structure should be clearly marked up.
2. Explicit use of absolute commands should be avoided.
3. Use of macros should reflect semantics and logical structure rather than physical layout.

These constraints have been introduced while discussing the global document structure. However they hold equally well when considering specific components of a document, such as mathematical expressions occurring within the document.

Format independent techniques of information capture. The above discussion also reveals the need for developing a framework for representing information in electronic documents independent of any single "display" method. TEX goes a long way in achieving this for mathematical documents. However since all the primitive operators used to achieve this are also available to the average user, TEX documents do not always conform to the constraint that information in a document be represented independent of formatting details. As these concepts become better understood, we need to work towards the development of a language for representing structured information in electronic documents. Some of this has already been achieved by SGML[4] [SGM86, Org90] and there is a clear need to carry over this work to cover mathematical documents as well. Development of such format independent techniques of information capture will allow us to provide alternative methods of accessing the same information structures.

[4] Standard Generalized Mark-up Language

Reading mathematics aloud

The previous section pointed out the need for high-level information capture independent of specific formatting techniques. Given high level information structures, we need to develop adequate techniques for accessing this information using alternative perceptual modalities such as audio. This section addresses the various questions that arise in developing an effective notational system for mathematics in the audio world. In order to do this, we first analyze how visual notation works and attempt to apply some of what we learn to the audio world.

Features of visual math notation. Traditional math notation fully exploits the two-dimensional nature of the visual tablet. It is therefore"not linear" but achieves conciseness by using subscripts and superscripts. It relies on the eye's ability to move quickly across the paper, and uses visual cues like delimiters of different sizes to cue these structured movements. Written mathematics is not "linear" in two different senses of the term, i.e., "space" and "time".

1. It uses a two-dimensional display.
2. It is not linear in time since the eye is able to move back and forth across the paper seemingly at will.

Both of these features are absent in traditional spoken mathematics. Spoken mathematics on tape is linear in time. In addition, it tends to be verbose since there is no standard way of alerting the listener to complicated syntactic constructions other than describing these verbosely (see Chang [Cha83], which is used as a guideline by RFB for reading mathematical texts). This can be directly attributed to the apparent lack in audio of the two-dimensional nature of the printed medium. This lack of two-dimensionality is overcome by the reader inserting words like "open quantity" and "raised to the quantity" in order to cue the listener to the presence of complex constructions. Use of such cues, though effective, causes the audio rendering to be necessarily verbose, making it difficult to grasp the essence of what is being conveyed. Mathematical notation relies on conciseness to express high-level concepts, and as the complexity of the expressions being handled increases, the resulting verbiage in the spoken equivalent of the written expression renders it practically unusable. We need adequate audio substitutes for these features of mathematical notation if we are to have any hope of effectively conveying mathematics using audio.

Spoken mathematics. The previous paragraph takes care to speak of *mathematics* in *audio* rather than merely referring to "spoken mathematics". This choice of terminology is intentional. One of the ways traditional spoken mathematics conveys complex inter-relationships between sub-expressions is to use prosodic cues within the speech. See the appendix, "Text taken verbatim from RFB's recording", for an explicit example. In order to achieve the full expressiveness of a human reading mathematics we need to use a lot of prosodic cues in the speech. Though computers of today can talk intelligibly they are still a long way from achieving this degree of expressiveness. We therefore need to augment the computer's speech ability by a well designed system of audio cues which can then be used to better convey extra-textual content when reading complicated expressions.

Multiple channels of audio. Non-speech audio cues can be synchronized with the speech output. Thus by using multiple channels of audio output, i.e., by having both speech and non-speech audio cues playing at the same time, and exploiting directionality of sound, we can offset the disadvantages resulting from not having a two-dimensional display. In fact the audio document is not restricted to a flat display, and proper use of audio cues can result in effective communication of complex constructs.

Browsing in an audio document. The previous subsection addresses the problems resulting from traditional audio documents being linear in space. This subsection in turn addresses the problems resulting from the fact that conventional recordings on tape have been linear in time.

A serious difficulty faced when listening to the recording of a complex expression is that the listener is forced to retain the entire expression. Thus comprehending spoken mathematics demands a longer attention span. In fact often it becomes impossible to remember the beginning of a complex expression by the time one has reached its end. This is clearly evinced by the reading of Faa de Bruno's formula presented in the appendix of this paper. Visual notation, by using different sized delimiters, different levels of subscripts, etc., cues these structured movements, and thereby allows the eye to move around the printed expression. Analogously we need to allow the listener to move around the expression and access parts of it at will. This will obviate the need for the listener to retain the entire expression in memory.

These structured movements can be performed using the high-level model for the expression that has been constructed at the recognition step. This is one of the major advantages of first recognizing the structure before generating audio renderings.

Conclusion.

This paper describes an audio view of (LA)TEX documents. Electronic typesetting source can be used to generate audio documents as well. In order to do this effectively, we need to recognize document structure from the electronic source. Audio renderings will then be driven from this high level structure. There is a need to develop notions of audio formatting analogous to the well understood notions of visual formatting. Finally we need to better understand how visual browsing works in order to build into the system the ability to provide the same functionality in the audio setting.

References

[Cha83] Larry A. Chang. *Handbook for spoken mathematics*. Lawrence Livermore National Laboratory, 1983.

[Knu84] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[Knu86] Donald Ervin Knuth. *TEX: The Program*. Addison-Wesley Pub. Co., Reading, Mass., 1986.

[Lam86] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley Pub. Co., Reading Mass., 1986.

[Org90] International Standards Organization. *Information Technology — SGML Support Facilities — Techniques for Using SGML*, 1990. Draft.

[Ram91] T. V. Raman. TEXTALK. *TUGboat*, 12:178, March 1991.

[SGM86] *Information Processing—Text and Office Systems—Standard Generalized Markup Language SGML*, October 1986. ISO 8879-1986 E.

Appendix: An example of complex math expressions

This appendix gives the TEXsource for a complicated mathematical expression, the transformed text generated by TEXTALK, and finally the text as read on the RFB recording of the same expression.

Note: This piece of mathematical text is taken from *Art of Programming, Vol. 1, 2nd Edition, Section 1.2.5, p. 50, Ex. 21* by Donald E. knuth.

The TEXsource

```
{\bf 21.} {\em [HM25]} (Faa di Bruno's formula.)
Let $D^k_x u$ represent the $k$th derivative of a function $u$ with
respect to $x$.  The ``chain rule'' states that $D^1_xw = D^1_u w
D^1_x u$.  If we apply this to second derivatives, we find $D^2_xw =
D^2_u w (D^1_x u)^2+D^1_u w D^2_x u$.
Show that the {\em general formula\/} is
$$D^n_xw =
\sum_{0\le j\le n}
\sum_{\scriptstyle k_1+k_2+\cdots+k_n=j
\atop {\scriptstyle k_1+2k_2+\cdots+nk_n=n
\atop  {\scriptstyle k_1,k_2,\ldots,k_n\ge0 }}}
D^j_u w \frac{n!}{k_1!{(1!)}^{k_1} \cdots k_n!{(n!)}^{k_n}}
{(D^1_x u)}^{k_1} \cdots {(D^n_x u)}^{k_n}.$$
```

The above piece of TEXcode was keyed in while listening to the reading of the expression on the RFB tape. This shows up an interesting fact about ordering of subscripts and superscripts. In this case the reader has read the superscript first, and the TEXsource reflects this in that it uses `D^1_x` rather than the more standard `D_x^1` as recommended in *The TEXbook*. At this time, there seems to be no reason to choose the reader's order of speaking the subscript after the superscript in the general case as against the order one would use if scanning the standard TEXusage linearly. In the case of Faa De Bruno's formula, the reader has used his interpretation of D_x^1 in making this choice. In the interest of being able to easily parse the TEXsource, we need to stick to either one of the two orderings when writing TEXdocuments. The order in which the subscripts and superscripts are eventually read can then be decided at a later stage in the audio formatting.

Text taken verbatim from RFB's recording

Exercise 21has a rating of cap h cap m 25
Faa de Bruno's formula.
Let D super k sub x of u represent the kth derivative of a function u with respect to x. The chain rule states that cap d super 1 sub x of w equals cap d super 1 sub u of w cap d super 1 sub x of u. If we apply this to second derivatives, we find that cap d super 2 sub x of w equals cap d super 2 sub u of w times the quantity cap d super 1 sub x of u quantity squared plus cap d super 1 sub u of w times cap d super 2 sub x of u. Show that the general formula is cap d super n sub x of w equals the summation from 0 less than or equal j less than or equal n of the summation over k sub 1 plus k sub 2 plus and so forth plus k sub n equals j, k sub 1 plus 2 k sub 2 plus and so forth plus n k sub n equals n, k sub 1 comma k sub 2 comma and so forth comma k sub n greater than or equal to zero The quantity being summed is
cap d super j sub u of w times the fraction n factorial over k sub one factorial times one factorial raised to the k sub 1 times and so forth times k sub n factorial times n factorial raised to the k sub n the entire fraction times
the quantity cap d super 1 sub x of u closed quantity raised to the k sub 1 times and so forth times the quantity cap d super n sub x of u closed quantity raised to the k sub n

Transformed text generated by TEXTALK

21. [HM25] (Faa di Bruno's formula.)
Let D super k sub x u represent the k th derivative of a function u with respect to x . The "chain rule" states that D super 1 sub x w = D super 1 sub u w D super 1 sub x u . If we apply this to second derivatives, we find D super 2 sub x w = D super 2 sub u w (D super 1 sub x u) super 2 + D super 1 sub u w D super 2 sub x u . Show that the general formula is
D super n sub x w = sum sub 0 less than or equals j less than or equals n sum sub k sub 1 + k sub 2 +

ellipses + k sub n = j and below that k sub 1 + 2 k sub 2 + ellipses + n k sub n = n and below that k sub 1 , k sub 2 , and so on , k sub n greater than or equals 0 D super j sub u w fraction with numerator n factorial and denominator k sub 1 factorial (1 factorial) super k sub 1 ellipses k sub n factorial (n factorial) super k sub n end of fraction (D super 1 sub x u) super k sub 1 ellipses (D super n sub x u) super k sub n .

The resulting formatted output

21. *[HM25]* (Faa di Bruno's formula.)

Let $D_x^k u$ represent the kth derivative of a function u with respect to x. The "chain rule" states that $D_x^1 w = D_u^1 w D_x^1 u$. If we apply this to second derivatives, we find $D_x^2 w = D_u^2 w (D_x^1 u)^2 + D_u^1 w D_x^2 u$. Show that the *general formula* is

$$D_x^n w = \sum_{0 \le j \le n} \sum_{\substack{k_1 + k_2 + \cdots + k_n = j \\ k_1 + 2k_2 + \cdots + nk_n = n \\ k_1, k_2, \ldots, k_n \ge 0}} D_u^j w \frac{n!}{k_1!(1!)^{k_1} \cdots k_n!(n!)^{k_n}} (D_x^1 u)^{k_1} \cdots (D_x^n u)^{k_n}.$$