www.elsevier.com/locate/carres

Carbohydrate Research 336 (2001) 1–11

CARBOHYDRATE RESEARCH

LINUCS: LInear Notation for Unique description of Carbohydrate Sequences

Andreas Bohne-Lang,[a] Elke Lang,[b] Thomas Förster,[a,c] Claus-W. von der Lieth[a,]*

[a]*German Cancer Research Centre, Central Spectroscopic Department (R0400), Im Neuenheimer Feld 280, D-69120 Heidelberg, Germany*

[b]*University of Applied Sciences Darmstadt, Department for Information and Knowledge Management, Schöfferstraße 1-3, D-64295 Darmstadt, Germany*

[c]*Current address: Tonbeller AG, Burgstraße 22, D-64625 Bensheim, Germany*

Received 26 March 2001; received in revised form 10 August 2001

Abstract

The use of proteomics databases has become indispensable for daily work of molecular biologists, but this situation has not yet been achieved for carbohydrate applications. One obvious reason is that existing data collections are only rarely annotated and no cross-linking to other resources exists. The existence of a generally accepted linear, canonical description for carbohydrates which can be readily processed by computers will enable efficient automatic cross-linking of distributed carbohydrate data collections by serving as a unique and unambiguous database access key. Various possibilities to derive a canonical notation are discussed. They can be divided into attempts that require structure description alone and alternatives that profit from the fact that a preferred graph direction (non-reducing to reducing end) exists within the structure. To open a fruitful discussion among glycoscientists a possible solution is presented where the reducing monosaccharide unit is selected as graph root and linkage information is used to define the priority of the various branches. A Web interface (http://www.dkfz.de/spec/linucs/) has been created that directly converts the commonly used extended representation of complex carbohydrates into the preferred canonical description or into its inverted form. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Carbohydrate sequence; Canonical description; Glycodatabase; Glycobioinformatics

1. Introduction

The use of proteomics databases has become indispensable for the daily work of the molecular biologist, but this situation has not yet been achieved for carbohydrate applications. Unfortunately, certain aspects of specification of carbohydrate nomenclature often make it laborious to find all references related to a certain carbohydrate in such general-purpose reference collections as Medline (PubMed). Even taking into account that the number of scientists working on various topics in the glycosciences is considerably smaller than the number of molecular biologists working with proteins and nucleic acids,[1] it is obvious that the acceptance of carbohydrate-related data collections is considerably lower in the community of glycoscientists than the acceptance that various proteomics data collections and tools receive by molecular biologists. Moreover, the opposite seems to become true. The CarbBank project,[2–4] the largest collection of carbohydrate-related references that

* Corresponding author. Fax: +49-6221-424554.
E-mail address: w.vonderlieth@dkfz.de (C.-W. von der Lieth).

had been built up during the 1980s and 1990s, entered shutdown mode in 1999 due to lack of funding. CarbBank used efficient algorithms to provide rapid access to references following the structure input of a query expressed in terms of the carbohydrate nomenclature. However, mere bibliographic information such as author(s), journal, and title can be displayed, but not complete abstracts. Compared to the intensive cross-linking and annotation of data and tools available for proteomics applications, CarbBank did not allow to find other important data for the compound of interest and to visualise them in a compact and well-structured representation.

Recently a few attempts have been described[5,6] aiming to create tools that link available information on glycans from various sources. GlycoSuite and BOLD are databases which currently are cross-linked with MEDLINE and SWISS-PROT/TrEMBL and contain annotated information extracted from scientific literature on glycoprotein-derived glycan structures. The company GlycoMind (http://www.glycominds.com) currently builds up a database that compiles information about glycoconjugate molecules, including their structures, functions, and interactions with other molecules.

In order to establish a simple link between various data sources, a unique, linear notation is required which can be directly derived from a given structure. Most of the genomic and protein-related data collections are linked using the sequence of the molecule of interest. However, topologies for complex carbohydrates significantly differ from the simple linear form describing genes and proteins: the number of residues occurring in Nature is much larger for carbohydrates. Whereas their sequences are considerably shorter (the average sequence length in CarbBank is about 6 residues) than those of proteins, each pair of monosaccharide residues can be linked in several ways, and one residue can be connected to three or four others (branching).

On the other hand, the known topologies of carbohydrates are much less complicated than those occurring in general in organic molecules, where various types of condensed ring systems, high order of branching and stereochemistry have to be considered. SMILES[7,8] is a linear chemical notation language which is used in many Web-based applications as a structure-based access key to link data of one specific compound contained in several distributed data collections. A unique SMILES[9] notation for each chemical structure is generated, regardless of the many possible equivalent descriptions that might be given as structure input.

Due to the physiological process of synthesis by Nature, all glycans exhibit two distinguishable functional groups termed the reducing and non-reducing end. The existence of a preferred direction defined by Nature is similar for other biological macromolecules, but does not exist for organic molecules in general. A detailed recommendation for carbohydrate nomenclature[10] is widely used by glycoscientists. However, the guidelines available for derivation of a hierarchy of the various branches are limited. Additional rules have to be applied to overcome these problems.

Several new and useful applications for glycosciences (e.g., CAZy,[11] 3D-Lectin (http://www.cermav.cnrs.fr/databank/lectine/) and *O*-GlycoBase[12] (see e.g. the collection of links at http://www.dkfz.de/spec/links/) have appeared on the Web during the last few years. They are mainly based on protein sequence information and unfortunately do not cross-link to other carbohydrate applications. The existence of a generally accepted linear, canonical description for complex carbohydrates, processable by computers with ease, will enable efficient cross-linking of carbohydrate data collections.

The aim of this work is to discuss various conceivable ways to generate a unique, linear description of glycans and to present our preferred solution as a proposal for a discussion among glycoscientists. It is not intended as a definitive and officially endorsed nomenclature system. A Web interface is available whereby anyone can produce a linear description for his/her structures of interest. Additionally we intend to distribute the source code (written in C) free, so that developers of carbohydrate-related databases and/or software can implement the description in their own applications.

2. Objectives and approach

Linear notations for molecular structure description are widely used in database applications, as computers can process linear strings readily and quickly. A unique notation, serving as access key, is required for rapid retrieval of data concerning one glycan contained in several distributed data collections. The IUPAC recommendation of the extended form to describe oligosaccharides (http://www.chem.qmw.ac.uk/iupac/2carb/38.html) is well-suited to serve as a basis for canonical description. The use of symbols consisting of three letters for monosaccharide residues is recommended. With appropriate locants and anomeric descriptors, long sequences can thus be adequately described in abbreviated form. A sequential naming approach is used, with the individual monosaccharide units abbreviated and the linkages between them indicated by citing the numbers of the connected atoms of the monosaccharide units. Branched structures are most often represented in a pseudo two-dimensional graph (See Example 1). Although IUPAC convention requires that the reducing group of the glycan structure be located at the right-hand end, the algorithm to be developed should not necessarily depend on this convention. IUPAC convention says that branched structures can be represented in a linear form by placing the branches inside square brackets. In a branched chain, the longest chain is regarded as the parent. If two chains are of equal length the one with lower number of locants at the branching point is preferred.

An algorithm to generate a linear, unique notation for description of complex carbohydrates should fulfil the following conditions:

1. Extended, non-graphic nomenclature to describe complex carbohydrates as recommended by IUPAC (http://www.chem.qmw.ac.uk/iupac/2carb/38.html) should be used as input.[†]
2. The resulting linear code should be closely related to the notations and abbreviations recommended by IUPAC.
3. The number of additional rules to define the priority of the branches should be as low as possible.
4. Extended nomenclature of complex carbohydrates should contain all information to apply the rules. No additional rules to define the hierarchy should be necessary.
5. Nomenclature should be applicable to all types of carbohydrates.
6. Nomenclature should be easily interpretable by glycoscientists.
7. Remaining unassigned linkage information should be tolerated during further processing.

One might argue that using full monosaccharide names will result in lengthy codes requiring a lot of disk space. This is of course true. Using for example a two-character code for the various monosaccharides and their modifications would indeed result in a much more compact code. However, no generally accepted two-character code for monosaccharides exists, so that the resulting notation could only be understood and transformed using a conversion list, and each newly found modification of a monosaccharide would have to be added to the list of encoded entries. This problem would probably result in inconsistencies between distributed data collections since it normally takes some time until a new abbreviation is generally known and accepted. Moreover, a two-letter code can handle only $35 \times 35 = 1225$ (alphabet plus numbers) differ-

[†] Non-carbohydrate substituents like OMe, OAc, phosphate, sulfate and peptides can be encoded in two ways. (a) as an extension to the monosaccharide units specifying the ring atom to which it is connected (e.g a-D-Galp4OMe, b-D-Glcp3SO3). (b) a linked residues (e.g. a-D-Galp-(3-0)-SO3, b-D-Galp (1-1)-ASN). Both implementations have advantages and disadvantages. Whereas the first implementation will not effect the hierarchy within the proposed linear notation, larger substituents like peptide chains are better encoded as connected residues. The main disadvantage to encode simple subsitutions as linked residues is that the hierarchy of the resulting notation is influenced and that the graphs become more complex so that they may loose their clarity. In our implementation of SWEET-II we handle (except for aminoacids) non carbohydrate substituents as extensions to the monosaccharide units.

ent codes. This number will definitively not be sufficient to encode all monosaccharides and their possible modifications.

Brevity of notation and economy of alphabet are not primary objectives of our approach. Many of the pitfalls in early attempts of linear notations in chemistry can be attributed to overuse of symbols and hierarchical rules in order to make the resulting code as compact as possible. Advances in computer hardware and software have made these restrictions obsolete. If for some reasons a compression of the notation is required, the conversion should be done automatically by the application program without affecting input and output procedures. Regarding the purpose of structure notation as the database access key, the focus should lie on a representation form that is semantically obvious and can easily be checked during all phases of processing, by machines as well as by scientists. This implies that it is not practical for use as a registry description or for use in extremely large bibliographic databases. For the sake of compactness, an appropriate data structure (inverted file or concordance table containing code and corresponding compact index) could be used internally by application programs or database management systems. Such a code describing a complete oligosaccharide is not well suited to be used for description and retrieval of substructures.

The great success of linear codes for nucleic acids and proteins is due to fact that these macromolecules themselves exhibit a linear structure allowing to align their sequences and are thus able to detect evolutionary and structural relationships. In contrast, for small organic molecules, no generally applicable code to describe similarities exists.[13] Many schemes are discussed on how to find structural relationships among the many hits normally resulting from high throughput experiments.[14] Concerning the description of molecular similarities, complex carbohydrates resemble more general organic molecules than they do nucleic acids and protein sequences, where no branches and ring closures can occur. Therefore, it is rather unlikely that one linear code can be derived which is able to describe structural identity as well as similarity.

Nevertheless, during the processing of complex carbohydrate structures by dividing them into substructures, a basis is established for substructure coding. Thus one structure parsing process can serve for derivation of a unique notation as well as for generating substructure codes.[15] An implementation of the latter can be tested via the user surface of SWEET-DB (http://www.dkfz-heidelberg.de/spec2/sweetdb/) In the following, however, we focus on outlining the principles of generating unique notations.

3. Ways to generate unique notations

Several approaches for generation of unique notations of complex carbohydrates are imaginable that can fulfil the aforementioned conditions (1)–(7). They can be divided into those which require topology description alone and those which make use of the fact that a preferred direction (non-reducing to reducing end) exists within the structure.

Approaches using structural information alone.—The first step for notations based on structural information alone is to define a hierarchy within the molecule. IUPAC recommendations allow the description of a unique, linear code for unbranched (sub)structures. Branched structures are encoded using square brackets. However, the guidelines available for deriving a hierarchy of the various branches are limited and insufficient. Terminating, biconnected and branched residues are easy to define by transforming structural information into a graph in which monosaccharides are represented by nodes and linkages are represented by edges. Monosaccharides connected to only one further residue form the end of a branch, those showing two connections are inner parts of a branch and those connected to more than two residues are branching points. On this basis, four possible approaches to derive a unique description will be outlined.

(1) The most simple approach, using only one sorting criterion in addition to IUPAC rules, is the generation of all possible combinations of linear codes, taking each terminating residue as the root. The resulting code with the lowest lexical order is taken as the

```
α-D-Glcp-(1→6)+
              |
         α-D-Manp-(1→6)+
              |         |
β-D-Manp-(1→3)+    β-D-Manp-(1→4)-β-D-GlcpNA-(1→4)-β-D-GlcpNA
                        |
         α-D-Galp-(1→3)+
```

Example 1. Glycan structure given in the extended representation of complex carbohydrates.

unique description. Further criteria, such as linkage type and branching degree, are not considered by this simple approach.

Example 1 outlines the proceeding for a *N*-glycan fragment.

Converting the extended representation to a labelled graph results in

```
[1]-+
    |
   [2]--+
    |   |
[3]-+  [4]-[6]-[7]
        |
   [5]-+
```

The graph contains four terminal residues: [1], [3], [5] and [7], one connecting residue [6], and two branching residues [2],[4].

Starting from each terminal residue node, all possible codes have to be generated to achieve exhaustive permutation. Using the graph notation and placing branches inside brackets as recommended by IUPAC, the following codes will result.

```
Terminal residue [1]
(a)  [1]-[2](-[3])(-[4](-[5])(-[6]-[7]))
(b)  [1]-[2](-[3])(-[4](-[6]-[7])(-[5]))
(c)  [1]-[2](-[4](-[5])(-[6]-[7]))(-[3])
(d)  [1]-[2](-[4](-[6]-[7])(-[5]))(-[3])

Terminal residue [3]
(a)  [3]-[2](-[1])(-[4](-[5])(-[6]-[7]))
(b)  [3]-[2](-[1])(-[4](-[6]-[7])(-[5]))
(c)  [3]-[2](-[4](-[6]-[7])(-[5]))(-[1])
(d)  [3]-[2](-[4](-[5])(-[6]-[7]))(-[1])

Terminal residue [5]
(a)  [5]-[4](-[2](-[3])(-[1]))(-[6]-[7])
(b)  [5]-[4](-[2](-[1])(-[3]))(-[6]-[7])
(c)  [5]-[4](-[6]-[7])(-[2](-[1])(-[3]))
(d)  [5]-[4](-[6]-[7])(-[2](-[3])(-[1]))

Terminal residue [7]
(a)  [7]-[6]-[4](-[5])(-[2](-[3])(-[1]))
(b)  [7]-[6]-[4](-[5])(-[2](-[1])(-[3]))
(c)  [7]-[6]-[4](-[2](-[1])(-[3]))(-[5])
(d)  [7]-[6]-[4](-[2](-[3])(-[1]))(-[5])
```

Lexical ordering is performed from left to right on residue codes in the respective positions according to ASCII code order, not according to the residue label numbers indicated above. All codes starting with residue [5] (α-D-Gal*p*) will exhibit lower lexical order than the other codes starting from the other three terminal residues (α-D-Glc*p* [1], β-D-Glc*p*NAc [7], β-D-Man*p* [3]). Here, only the monosaccharide name is taken to derive a hierarchy for the four codes starting from each terminal residue. The code associated with the linear graph [5] [4](-[2](-[3])(-[1]))(-[6]-[7]) is preferred. Linkage information and branching degree would not be taken into account even if remaining ambiguities could not be solved.

Such an attempt is well suited for computer applications since the various codes can be generated and sorted easily. A disadvantage is that it will be laborious for glycoscientists to generate all possible codes and identify the one with the property of uniqueness.

(2) A reduction of the number of codes to be generated can be achieved applying a second criterion: At each branching point, the hierarchy of the attached branches is given using linkage information. Linkage types are sorted according to atom numbers, e.g. $(1\to3)<(1\to4)<(1\to6)$. Again, the resulting code with the lowest lexical order (in this case, determined using linkage information) is used as unique description. For the glycan structure of example 1, the following codes result:

```
(a)  [1]-[2](-[4](-[6]-[7])(-[5]))(-[3])
(b)  [3]-[2](-[4](-[6]-[7])(-[5]))(-[1])
(c)  [5]-[4](-[6]-[7])(-[2](-[3])(-[1]))
(d)  [7]-[6]-[4](-[5])(-[2](-[3])(-[1]))
```

Again the code starting at terminating residue [5] (α-D-Gal*p*) will be the code with lowest lexical order since the code starting with [5] α-D-Gal*p* exhibits a lower lexical order than α-D-Glc*p* [1], β-D-Glc*p*NAc [7] or β-D-Man*p* [3]). In such a way only one description for each terminating residue has to be created.

(3) Alternatively, the branching point exhibiting maximal number of connections (highest degree of branching) is taken as root. Again the hierarchy of the attached branches can be defined by their linkage information. If two or more branching points possessing equal maximal numbers of connections exist (this will hold quite often for *N*-glycans), the code is generated for each of them applying linkage information to establish a hierarchy within the code.

Example 1 has two branching points resulting in the following codes:

```
[2]: [2](-[4] (-[6]-[7]) -[5]))(-[3])(-[1])

[4]: [4](-[6]-[7]) (-[5]) (-[2] (-[1]) (-[3]))
```

The code starting with [2] (α-D-Man*p*) will be taken as the unique description. The number of codes to be generated is considerably reduced (normally only two or three codes have to be created) and thus this approach would be easier to adopt for scientists.

(4) Another approach would be to find the longest path within a graph which can be defined as primary chain. If two or more chains show the same length, the one with the highest lexical order will be preferred. For the molecule given in Example 1 the following twelve codes will result.

Chain length 3	Chain length 4	Chain length 5
[1]-[2]-[3]	[1]-[2]-[4]-[5]	[1]-[2]-[4]-[6]-[7]
[3]-[2]-[1]	[5]-[4]-[2]-[1]	[7]-[6]-[4]-[2]-[1]
	[3]-[2]-[4]-[5]	[3]-[2]-[4]-[6]-[7]
	[5]-[4]-[2]-[3]	[7]-[6]-[4]-[2]-[3]
	[5]-[4]-[6]-[7]	
	[7]-[6]-[5]-[4]	

Among the four codes containing five residues, the one with termination residue α-D-Glc*p* [1]-[2]-[4]-[6]-[7] will exhibit the lowest lexical order. The complete code would look like:

```
[1]-[2](-[4](-[6]-[7])(-5))(-[3])
```

Alternatively, the chain with higher degree of branching can be defined as primary chain. The same procedure has to be repeated for each branch containing one or more branching points. Finally, another rule has to be applied to select one of the possible notations of the primary chain. It is obvious that such an approach would definitively require more rules than the one discussed above. Since one of our basic requirements is that the number of rules to define the priority of the branches should be as low as possible, this approach can be neglected.

A general problem of the discussed approaches is that the resulting linear code may not be related in an obvious way to commonly used notations for carbohydrates in publications. Therefore it might be laborious to identify the rules of their mutual depiction.

Approaches that make use of the preferred direction.—As with the description of nucleotide and protein sequences, a preferred direction defined by Nature exists in complex carbohydrates. According to IUPAC convention, the reducing end, which is a unique feature in most carbohydrate structures (exceptions are mentioned in the following subsection), is always located at the right-hand side of the notation. Thus, one of the non-reducing residues or the reducing end, which also represents the attachment site to proteins or lipids, can be defined as the root for encoding the structure. In addition three possible approaches making use of the preferred direction will be outlined.

(1) Defining the reducing residue as root, the hierarchy of the branches can be readily defined using rules predefined by Nature. Although one monosaccharide can be connected maximally to four others, a hierarchy of the branches can be easily established by simply using linkage information. $(2 \rightarrow 1)$ connections receive higher priority than $(3 \rightarrow 1)$ linkages. If one unknown linkage exists, it will be assigned

the lowest priority. Only in cases where two or more unassigned linkages exist, an additional rule has to be applied. An algorithm that is easy to implement will generate codes for all possible branches and then sort them in lexical order. Therefore it seems to be obvious to use lexical ordering to remove ambiguity which occurs when two or more unassigned linkages exist. Only one linear code results, which will show the monosaccharide forming the reducing end at the beginning of its notation. In order to compare the code to the extended representation of complex carbohydrates in publications, one would have to read from right to left. Since this may be inconvenient, the resulting code may be reverted. The implementation of this code will be discussed in detail in section 4.

(2) In this attempt the terminating unit of the branch exhibiting the highest priority is identified defining the reducing unit as root and evaluating priority according to linkage information. In a subsequent step, the terminating unit of the assigned branch is defined as root and a new notation is generated using linkage information to define the hierarchy of the branches. The resulting notation can be directly compared to the extended representation reading from left to right. For example 1, residue [5] (($1 \rightarrow 3$) α-D-Gal*p*) exhibits the highest priority. The resulting code would be [5]–[4](–[6]–[7])(–[2](–[3])(–[1])).

(3) A limited number of codes has to be generated when taking each non-reducing residue as root and using linkage information to establish the priority of the various branches. Lexical sorting of the resulting codes will define a unique one. The branch exhibiting the highest priority will appear at the beginning of the code and can be easily compared to the commonly used representation of complex carbohydrates. For example 1 the resulting three codes are:

```
(a) [1]-[2](-[4](-[6]-[7])(-[5]))(-[3])
(b) [3]-[2](-[4](-[6]-[7])(-[5]))(-[1])
(c) [5]-[4](-[6]-[7])(-[2](-[3])(-[1]))
```

Lexical sorting defines the code starting from residue [5] as unique one.

(4) Cooper et al.[5,6] also implicitly relied on the existence of a preferred orientation and applied four additional rules resulting in a canonical description of branched glycan structures. Lengths of the various branches, their degree of branching, lexical order of the termination residue and the lowest terminal linkage are applied to define the hierarchy. Since the number of rules required for this approach is considerably higher than for the approaches already discussed, which only need one additional rule (sorting criterion), we do not discuss in detail its consequences for the implementation of an appropriate algorithm.

A strong advantage of approaches making use of the preferred direction within carbohydrate structures is that the resulting linear code can be easily related to the commonly used notations in publications, although it might take some time to get accustomed to reading it.

Special cases.—($1 \rightarrow 1$) linked disaccharides (such as sucrose) and macrocyclic molecules (e.g. cyclodextrins) do not exhibit a preferred orientation within the molecule, and additional rules have to be applied to be able to create a unique description.

Whereas for ($1 \rightarrow 1$) linked disaccharides lexical sorting will remove ambiguity, a unique description of such macrocyclic molecules as cyclodextrins requires two additional rules. IUPAC recommendations do not suggest any notation for an alphanumeric, non-graphical representation for macrocyclic carbohydrate structures which might be suitable for computerized applications. CarbBank uses a notation (see below) where macrocyclic rings are indicated by the keyword ‘Cyclic’ followed by linkage information concerning the reducing side. The end of the cycle is indicated by a linkage information of the reducing side (normally 1) and a hyphen. Residues to be repeated are defined by brackets, and the size of the macrocycle is indicated by the number of repeats. Attached branches can be added using the normal extended description. Fig. 1 depicts a spatial structure of the example given next.

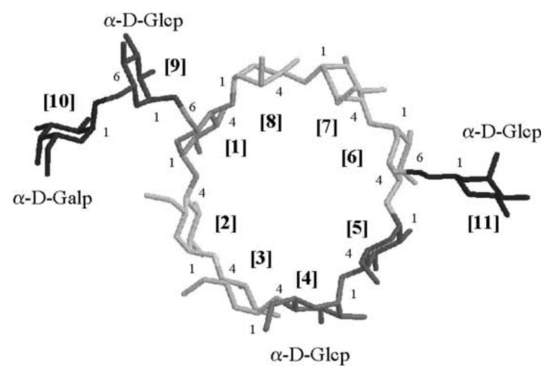

Fig. 1. 3D representation of a cyclodextrin with eight $(1\rightarrow4)$-linked α-D-Glc*p* units in the macrocycle and two attached branches. The symbolic description of the residues is given in square brackets. Linkage information for the glycosidic linkage is displayed next to the corresponding C atom. The 3D structure was generated with the aid of the SWEET-II[13] web interface (http//www.dkfz.de/spec/sweet2/).

sugar residues are attached. In principle, an exhaustive generation of all possible linear codes and sorting according to their lexical order will produce a unique description. Nevertheless, this procedure may be quite laborious for humans. Therefore, we suggest defining the end residues of the attached branches as starting points. Taking linkage information into account, one code for each attached branch will result. If two or more branches are attached to the macrocyclic structure, the unique one will be identified comparing their lexical order.

For the given example (see Fig. 1) using the indicated graph notation the two resulting codes will be:

```
(a) [10]-[9]-[cyclic]-[1]-[2]-[3]-[4]-[5]-[6](-[11])(-[7]-[8]-[cyclic])
(b) [11]-[cyclic]-[6]-[7]-[8]-[1](-[10]-[9])(-[2]-[3]-[4]-[5]-[cyclic])
```

Example of a macrocyclic structure having two different attached side chains, as described in the CarbBank notation:

Description A exhibits a lower lexical order ([10] = α-D-Gal*p* < [11] = α-D-Glc*p*) and will be defined as the unique code.

```
                α-D-Galp-(1→6)-α-D-Glcp-(1→6)+
                                             |
α-D-Glcp-(1→6)+                     α-D-Glcp-(1→{4)-α-D-Glcp-(1}3-4)-α-D-Glcp-(1→
              |                              |
             α-D-Glcp-(1→{4)-α-D-Glcp-(1}2→4)+
              |
     Cyclic→4)+
```

For derivation of a unique linear description, an additional marker has to be introduced to indicate ring closures. The trivial case, which occurs quite often in practice, is when all *n* monosaccharides and linkages constituting the macrocyclic structure are equal. The same unique description will result, no matter which residue will be chosen as starting point by the algorithm. In the case of different monosaccharides and/or linkages, all *n* possible codes have to be generated and the unique one can be identified on base of lexical order and linkage information. The preferred direction within the macrocycle is given by the linkage information. This does not hold for macrocylic structures to which branches of

Currently the generation of a unique linear notation for macrocycles is not implemented in the LINUCS code.

4. Implementation

Comparing the various possible approaches to define a unique, linear notation for complex carbohydrates and taking into account the conditions (1) to (7) already specified, we favor the kind of approach that makes use of the preferred direction defining the reducing residue as root.

In this implementation, generation of a unique notation is derived in two steps. First a semantic transformation of the carbohydrate graph is performed applying syntax rules given below. The result is a framework that resembles nested loops in, for instance, C-language source code (see indented form given below example C). This framework can be easily transformed into a linear description-containing nested brackets (see below example C) by collapsing the multi-line indented structure to a linear form. In a second step the priority rules as discussed above are applied to create a unique notation. This two-step procedure has the advantage that the semantic transformation has to be done only once and that it is simple to implement and to test the various hierarchy rules. Here we give only the syntax rules applied and do not show the frameworks they produce. All linear representations given below are codes where the preferred hierarchy rules (reducing monosaccharide unit is selected as graph root and linkage information is used to define the priority of the various branches) are applied as well.

Semantic transformation.—Processing the structure graph (consisting of monosaccharide names, linkage information and connecting edges) starts with labelling residues from the uppermost left one, going forward according to the following scheme:

```
     3
 1   +   4
     2
```

This rule has been chosen arbitrarily, but must be applied consistently from then on.

Applied syntax rules:

(1)	S -> [ ] [Z] {A}	root symbol
(2)	A -> AA	element of the same hierarchy level
(3)	A -> [K] [Z] {A}	element of a lower level
(4)	A -> epsilon	zero element - termination symbol
(5)	K -> (E-E)	linkage, reading direction from left to right
(6)	K -> (E+E)	linkage, reading direction from right to left
(7)	Z -> saccharide name	monosaccharide (e.g. "a-D-Manp")
(8)	E -> linkage atom number	(normally 1, 2, 3, 4,5 or 6)

Applying hierarchy rule to the transformed graphs.—Three examples explain the procedure:

(a) Simple linear saccharide (extended nomenclature). The notation of monosaccharides is similar to IUPAC nomenclature, except that Greek symbols have been replaced by appropriate Latin letters (a instead of α). The italic pyranose designations are in roman type, and connecting arrows are replaced by hyphens.

α-D-Man*p*-(1 → 4)-β-D-Man*p*-(1 → 2)-α-D-Man*p*

will be transformed by rules 1, 3, 4, 5, 7, and 8 into

[ ][a-D-Man*p*]{[(2 + 1)][a-D-Man*p*]{[(4 + 1)][a-D-Man*p*]{}}}

or reverse notation:

[ ][a-D-Man*p*]{[(1 → 4)][b-D-Man*p*]{[(1 → 2)][a-D-Man*p*]{}}}

(b) Simple branched structure

```
b-D-Galp-(1-6)+
              |
        a-D-Manp-(1-4)-b-D-GalpNAc
              |
a-D-Galp-(1-4)+

[][b-D-GalpNAc]{[(4+1)][a-D-Manp]{[(4+1)][a-D-Galp]{}[(6+1)][b-D-Galp]{}}}
```

The result can be displayed in indented form:

```
[][b-D-GalpNAc]{
  [(4+1)][a-D-Manp]{
    [(4+1)][a-D-Galp]{}     \ (4 1) is less than (6 1)
    [(6+1)][b-D-Galp]{}     /
    }
  }
```

A short outline of application of syntax rules for semantic transformation illustrates how the algorithm works.

The monosaccharide β-D-GalpNAc forms the root for the oriented graph. It is connected to α-D-Manp by a 4-1-linkage. Residue α-D-Manp constitutes a branching point linked to α-D-Galp and β-D-Galp. Taking linkage information into account, the (3-1) linked branch will get a higher priority than the (6-1) linked one.

(c) *N*-Glycan with three branches

```
          α-D-Manp-(1→2)-α-D-Manp-(1→6)+
                                       |
                              α-D-Manp-(1→6)+
                                       |       |
         α-D-Manp-(1→2)-α-D-Manp-(1→3)+   β-D-Manp-(1→4)-β-D-GlcpNAc-(1→4)-β-D-GlcpNAc-(1→4)-Asn
                                               |
α-D-Manp-(1→2)-α-D-Manp-(1→2)-α-D-Manp-(1→3)+
```

Resulting linear code using the reducing end as root:

```
[ ][Asn]{[(4+1)][b-D-GlcpNAc]{[(4+1)][b-D-GlcpNAc]{[(4+1)][b-D-Manp]{[(3+1)][a-D-Manp]{[(2+1)]
[a-D-Manp]{[(2+1)][a-D-Manp]{}}}[(6+1)][a-D-Manp]{[(3+1)][a-D-Manp]{[(2+1)][a-D-Manp]{}}[(6+1)]
[a-D-Manp]{[(2+1)][a-D-Manp]{}}}}}}}
```

The result can be displayed in indented form (see following page).

5. Summary and discussion

One obvious reason that the use of glycorelated databases has not yet become indispensable for daily work of glycoscientists is the fact that existing data collections are only rarely annotated and exhibit no cross-linking to existing other resources. The existence of a generally accepted linear, canonical description for carbohydrates which can easily be processed by computers will enable efficient automatic cross-linking of distributed carbohydrate data collections by serving as a unique and unambiguous database access key.

A comprehensive discussion of the various possibilities for deriving a canonical, linear description for complex carbohydrates is presented here to initiate a fruitful discussion among glycoscientists. The main requirements for the resulting code are that it is semantically obvious, close to commonly used IUPAC nomenclature and that it can easily be checked during all phases of processing, by machines as well as by scientists. As with the description of nucleotide and protein sequences, a preferred direction defined by Nature exists in complex carbohydrates. The reducing monosaccharide unit is a unique feature in most carbohydrate structures, and thus it is a decision driven by Nature to select it as the root for encoding complex carbohydrates. Any set of rules as discussed here is in principle capable of creating the required canonical notations. However, we have shown that there are good reasons for selecting the reducing monosaccharide unit as the root. Using basic IUPAC recommendations of the extended description of oligosaccharides, only ordering of the resulting code by linkage information is required to directly create a canonical notation for all types of complex carbohydrates. Rare exceptions are some special cases for macrocyclic structures. This implies that, for branched chains, one criterion fewer (one instead of two) is necessary as recommended by IUPAC (longest chain as parent, if two chains of equal length exist, the one with lower locants at the branching point is preferred).

The resulting code can be easily decoded and encoded by humans and fulfils most of the conditions demanded in the objectives section. Only for molecules without a reducing

```
[][Asn]{
      [(4+1)][b-D-GlcpNAc]{
            [(4+1)][b-D-GlcpNAc]{
                  [(4+1)][b-D-Manp]{
                        [(3+1)][a-D-Manp]{
                              [(2+1)][a-D-Manp]{
                                    [(2+1)][a-D-Manp]{}
                              }
                        }
                        [(6+1)][a-D-Manp]{
                              [(3+1)][a-D-Manp]{
                                    [(2+1)][a-D-Manp]{}
                              }
                              [(6+1)][a-D-Manp]{
                                    [(2+1)][a-D-Manp]{}
                              }
                        }
                  }
            }
    }
}
```

unit (such as cyclodextrins) have additional rules to be applied. Brevity of notation and economy of alphabet are not primary objectives of our approach. If for some reasons a compression of the notation is required, the conversion should be done automatically by the application program without effecting input and output procedures.

A Web interface (http://www.dkfz.de/spec/linucs/) has been created which directly converts the commonly used extended representation of complex carbohydrates into the preferred canonical description or into its inverted form. Thus anyone can test the algorithm using his/her molecules of interest. Additionally we will distribute the source code (written in C), so that software developers can easily implement our algorithm within their own application. Common use in many applications and data collections will be the key to success for every nomenclature. Our intention is to initiate a broad and fruitful discussion by making these solutions generally available, in the hope that a generally accepted notation will result. Comments and criticisms are welcomed.

References

1. Hardy, B.; Wilson, I. *Glycoconj. J.* **1996**, *1*, 865–872.
2. Albersheim, P. *Glycobiology* **1991**, *1*, 113.
3. Doubet, S.; Bock, K.; Smith, D.; Darvill, A.; Albersheim, P. *Trends Biochem. Sci.* **1989**, *14*, 475–477.
4. Doubet, S.; Albersheim, P. *Glycobiology* **1992**, *2*, 505.
5. Cooper, C.; Wilkins, M.; Williams, K.; Packer, N. *Electrophoresis* **1999**, *20*, 3589–3598.
6. Cooper, C.; Harrison, M.; Wilkins, M.; Packer, N. *Nucleic Acids Res.* **2001**, *29*, 332–335.
7. Weininger, D. *J. Chem. Inform. Comp. Sci.* **1988**, *28*, 31–36.
8. Weininger, D. *J. Chem. Inform. Comp. Scien.* **1990**, *30*, 237–24.
9. Weininger, D.; Weininger, A.; Weininger, J. *J. Chem. Inform. Comp. Sci.* **1989**, *29*, 97–101.
10. IUPAC-IUBMB, Nomenclature of Carbohydrates, *Carbohydr. Res.* **1997**, *297*, 1–92.
11. Coutinho, P.; Henrissat, B. In *Carbohydrate-active enzymes: an integrated database approach*; Gilbert, H. J.; Henrissat, G. D. B.; Svensson, B., Eds.; The Royal Society of Chemistry: Cambridge, 1999.
12. Hansen, J.; Lund, O.; Rapacki, K.; Brunak, S. *Nucleic Acids Res.* **1997**, *25*, 278–282.
13. Radíc, A.; Zupan, J. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 550–560.
14. Estrada, E.; Molina, E. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 791–797.
15. Bohne, A.; Lang, E.; von der Lieth, C. W. *J. Mol. Model.* **1998**, *4*, 33–43.