In *International Workshop on Computing With Biomolecules*, E. Csuhaj-Varjú, R. Freund, M. Oswald and K. Salomaa (Eds.), 27 August 2008, Wien, Austria, pp. 75–86, From: Austrian Computer Society, 2008.

AN EXTENDED DOT-BRACKET-NOTATION FOR FUNCTIONAL NUCLEIC ACIDS

Effirul I. Ramlan Klaus-Peter Zauner
School of Electronics and Computer Science
University of Southampton, SO17 1BJ, United Kingdom
Email: {eir05r,kpz}@ecs.soton.ac.uk

Abstract
Functional nucleic acids are an attractive substrate for molecular computing. A nucleic acid molecule is a linear chain of covalently bound building blocks assembled in arbitrary order from a set of typically four nucleotides. Certain pairs of nucleotides weakly attract each other through short-range electrostatic interaction and, accordingly, complementary sequences of nucleotides can bind to each other. The complementary stretches of nucleic acids that attract each other can be part of two different molecules or two parts of a single molecule. Binding within a single molecule leads to a folding of the linear chain. This so called secondary structure is of great importance for the function of nucleic acids.

The present paper is concerned with the representation of this secondary structure. We propose an extension for the syntax of the standard dot-bracket notation to increase its convenience and expressive power for both its use to communicate nucleic acid secondary structures among humans and machines. The extensions reflect our own requirements for the representation of nucleic acids for molecular computation, but should be useful for functional nucleic acids in general.

1. Computational Nucleic Acid Enzymes

Organisms have powerful and enviably efficient information processing capabilities. To a large extent these capabilities are conferred by macromolecules and their specific properties. The existence of these natural information processing architectures demonstrates that computing based on physical substrates that are radically different from silicon is feasible. Accordingly, the potential of biomolecules as a computing substrate in artificial devices has been investigated for over three decades [15]. In nature proteins appear to play the preeminent role as molecular computing substrate. At the present state of technology, however, two other classes of biomolecules are more amenable to applications in man-made information processing architectures: deoxyribonucleic acids (DNA) and ribonucleic acids (RNA). The former offers a more limited conformational flexibility and concomitant less functionality, while the latter is

less stable and requires more careful laboratory techniques. The existing body of work on using nucleic acids for information processing can be grouped under three concepts:

Covalent Concept: Early proposals for the use of DNA in computing were inspired by the discovery of DNA's role in the storage of inheritable information and the astounding information density that can be achieved with molecular encoded data. All of these concepts require the formation and cleavage of specific covalent bonds which would require custom-designed proteins and are for this reason not practical (cf., e.g., [8])

Complement Concept: In [1] the influential suggestion to use arbitrary nucleotide sequences and the hybridisation with their complementary sequences instead of covalently linked individual bases was made and a practical demonstration of this approach was given. This idea moved the burden of recognising tokens of information from proteins (which up to now cannot be designed for purpose) to the self-assembly of short nucleotide sequences which can be designed with desired self-assembly properties and can be synthesised with ease.

Conformational Concept: Whereas in the above two concepts the conformational flexibility of the nucleic acids is irrelevant or even undesirable, more recently computing concepts that exploit the change in conformation a nucleic acid undergoes upon hybridising with another nucleic acid molecule have been developed [9, 10].

A recent introduction to the conformational concept of information processing with nucleic acids is available in [11]. The three-dimensional conformation of a nucleic acid is largely determined by the secondary structure, i.e., the intramolecular binding among complementary sections of its nucleotide sequence. The sequence itself is typically for the most part not as important for the function as the secondary structure it assumes. For a given secondary structure there is generally a large variety of nucleotide sequences that it will fold into.

In designing sets of nucleic acids for information processing one typically has a desired secondary structure and additional local constraints on the sequence. For example, a binding side for an effector molecule (i.e. a nucleic acid that will affect the activity of a functional nucleic acid) may be required to be complementary to a sequence released in a preceding step. Software tools that, given a secondary structure, can generate a suitable sequence candidate likely to fold into the desired structure are available [2, 3, 5]. It would be desirable to have a convenient representation of secondary structures together with sequence constraints and special properties of regions of the sequence. Ideally a single representation should support the communication between humans and software.

2. Representation of Nucleic Acid Structure

A widely used method to denote RNA secondary structure is the dot-bracket-notation or parenthesis format introduced by Hofacker et al. [2]. It uses matching parenthesis and dots to denote paired and free bases, respectively. Fig. 1 illustrates the notation for a short RNA sequence.

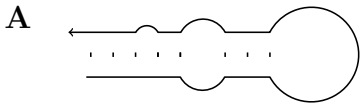


Figure 1: A sample secondary structure of an RNA molecule is shown (A) together with the notation of its sequence (B) and its structure in dot-bracket form (C).

The dot-bracket-notation has the advantage that a string denoting the secondary structure of a nucleic acid is of the same length as the string denoting the nucleotide sequence with a single character for each nucleotide. The two strings can be aligned to show the secondary structure features along the nucleotide sequence (Fig. 1B and C). In molecular biology one typically has a given (discovered) sequence and is interested in its folding properties. The dot-bracket notation reflects this mode of operation. As indicated above, in molecular computing applications it is common that the secondary structure is of importance, but the detailed sequence that yields the structure is over large stretches arbitrary. In such a scenario the dot-bracket notation is often cumbersome, it leads to large expressions with information that is partially obscured for human readers, because it would require counting identical characters. While the sequence is typically arbitrary in most positions, as long as the structure of the molecule is preserved, nucleic acids with functional properties, such as catalytic activity, often require specific bases in a few positions. If in a few places the nucleotide sequence (i.e. the primary structure) is given, two strings are required: one to specify the structure in dot-bracket-notation and a second string to represent the type of the immutable nucleotides. Furthermore, for communicating structural features among humans a two-dimensional rendering of the one-dimensional dot-bracket notation is often desirable. It would be convenient if specific features in the sequence could be communicated to the rendering software.

To alleviate these issues we use an extended dot-bracket-notation that is particularly suitable to denote functional nucleic acids where the primary characteristic is the secondary structure and not the sequence. In doing so we were aiming at:

- backward compatibility to the dot-bracket notation
- use of familiar and mnemonic conventions
- single character operators
- the possibility to describe sets of interacting molecules
- flexibility to chose expressions according to application
- support for rendering with and without colour

Achieving these aims comes at the price of giving up the equivalence of the length between the secondary structure specification and the sequence. On the other hand, the extended notation is often more compact and capable of describing, in a single string, a group of RNA molecules where each molecule varies in length, sequence, and conformation.

The extended dot-bracket-notation introduces several new symbols which fall into four different categories:

Scoping symbols group sections of the notations. Square brackets "[ ]" are used for grouping range of base positions. Curly braces delimit alphanumerical parameters for operators and also limit sets of constraints placed on the choice of base permitted for a particular sequence position. Curly braces can optionally be used to delimit the numbers specifying repetitions.

Operator symbols associate a property with the preceding base position, or grouped range of positions. The properties are mostly used for graphical rendering of the the structure (`_,$,~,@`), but also to mark binding sites for inter-molecular binding (`+`). An operator symbol is always followed by a parameter.

Constraint symbol restrict the possible bases that may be present at the preceding base position. The two constraint symbols are a colon (`:`) which restricts the preceding position to be equal to the base or set of bases that follow it, and a hat (`^`) that restricts the preceding position to differ from the base or set of bases that follow it.

Special symbols are available to express features which cannot be expressed with the above elements. At present only the `%`-symbol is defined, it marks a cleavage-point where the RNA sequence may be hydrolysed.

An overview of the new symbols introduced in the extended dot-bracket-notation is provided in Tab. 1. With these enhancements the extended dot-bracket notation can carry a lot more information about an RNA structure than the standard notation while generally leading to a more compact description. However, the translation from the extended dot-bracket-notation to the standard notation is trivial. This is important as the computational tools for nucleic acids secondary structure have adopted the dot-bracket-notation as their input–output channels [6]. A translation from the standard notation to the extended dot-bracket-notation can of course not make much use of the richer syntax of the extended notation. It is also generally not required, as the standard notation is a valid subset of the extended notation. Nevertheless, such a translation may be useful to arrive at shorter representations as shown for small examples in Tab. 2. Any symbol that can occupy a base position in a sequence (i.e., `.,(,),A,U,G,C,T,...`) may be followed by a positive integer value n to denote n repetitions of the symbol. This run-length notation is particularly convenient for manually entering secondary structure descriptions. The downside is that the run-length notation can obscure structural motives which may be recognised more readily in the standard notation. A considered use of the repetition parameter will maximise readability, whether by reducing the length of the representation or by deliberately breaking runs of parenthesis into sections that match. For example, `(3.2(3.4)6` represents a stem-loop with bulge. The same structure could also be written as `(3.2(3.4)3)3`. The latter is longer, but preferable nevertheless, because the base-pairing of the two helices is emphasised by breaking the run of six closing parenthesis into two groups of three closing parenthesis each. This example also illustrates that in the extended dot-bracket-notation there

Table 1: New symbols introduced in the extended dot-bracket-notation

	Description	Usage	Comment
`[ ]`	Grouping of base positions	`[.8]@{label A}`	Eight unbound bases marked as "label A".
`{ }`	Parameter delimiter		see example above
`{ }`	Set delimiter	`.:{A,C}`	A single unbound base that can be either A or C.
`{ }`	Repetition delimiter	`A{10}`	Always optional.
`_`	Line width	`((([.5]_1)))`	Stem-loop structure with bold loop
`$`	Colour	`(3[.2]$1(3.4)3)3`	A buldge in red.
`~`	Line decoration	`.24~1(3.3)3`	Binding site marked as crinkled line.
`@`	Annotation marker		See first row.
`+`	Multi-molecule binding	`(24+1(3.3)3`	Sticky end of 24 bases, will bind to site marked 1 on other molecule.
`:`	Base assignment	`)):A`	Two binding bases, the second one of which is A; See also set delimiter.
`^`	Base exclusion	`(((..^U.)))`	Stem loop where the central base in the loop is not a uracil.
`%`	Clevage point	`(((..%..(((`	Between bases, i.e., not a base position.

is no unique string to describe a given structure. The equivalence of two structures denoted in the extended form, however, can be established by translating both into the standard form, which is easily accomplished.

Specifications for individual base positions, repetitions of these, as well as groups (marked by square brackets) of individual positions and repetitions can be arguments for operators. The operator follows its argument and precedes its parameters. More than one operator/parameter combination may follow an argument and all will be applied to the argument. Table 3 provides a few examples of operator use—some of them taken from the structures rendered in Figs. 3 and 4. Note that in all cases the argument itself, which precedes the operator, is not shown.

The acceptable parameters that follow an operator and their semantics are not specified by the

Table 2: The extended dot-bracket-notation allows for run-length encoding to achieve a compact representation

Standard notation	Extended notation	Part of Fig.
`...(((((....)))))..)))))))))`	`.3(5.4)5.)8`	3A
`(((....)))....((((((((`	`(3.4)3.5(8`	3B
`(((((....)))))..)))))))).`	`(5.4)5.)8.`	4A
`..)).)))........`	`.2)2.)3.8`	4C

Table 3: Sample usage of operators

Notation	Description	Fig.
`~2@{shift region 1}`	Apply decoration type 2 to the preceding region and label it as "shift region 1".	3A
`+3~2`	The preceding argument binds externally with another molecule at the region marked "3" and is drawn with decoration type 2 ("cross").	3C
`+2_1@{OBS2+EFF2}`	The argument (not shown) binds with another molecule at the region marked "2", draw the binding region in bold (line thickness 1) and label it as "OBS2+EFF2"	4C
`+1_1${Red}@{node001}`	The preceding argument binds externally with another molecule at the region marked "1", render the argument with line thickness 1 in red and label the region as "node001"	-
`~1_2${blue}`	Combination of drawing parameters applied to the preceding argument resulting in a strong bold (thickness 2) crinkled line (type 1) in blue color.	-
`+{SITE1}~1_1@{match}${red}`	The preceding argument binds externally with another molecule at the region marked as "SITE1". This region is rendered using crinkle line with the thickness value 1, and colored in red. The region is labeled "match".	-

notation. A rendering program, for example, may accept a predefined color number, an explicit color name, or a hexadecimal RGB value. The corresponding operator with parameters would be `$1`, `${red}`, and `${#FF0000}`. An overview of the extended notation is provided in Fig. 2. For clarity, three of the nonterminal symbols occurring in the syntax graph are not shown in Fig. 2. The non-terminal *digit* stands for a single digit in the range from 0–9. The non-terminal *alpha* stands for a single character from either the range a–z, or A–Z, or a dash (-), underline (_), or space ( ). The non-terminal *base* stands for any one of (A,U,G,C,T,X,N) in upper or lower case.

As can be seen in Fig. 2, a single string in the extended dot-bracket-notation can denote more than one molecule (cf. *RNA_string* in Fig. 2). The limitations in expressing interactions among multiple molecules in the standard notation was one of the factors motivating the present work. An example with a pair of molecules that bind in two different regions of equal length will illustrate the difficulty of using the standard notation in such cases:

```
((((((.....((((((......))))))...((((....))))...((((((  &
))))))........(((((....((((....))))....))))))...))))))
```

The two lines represent two different molecules, separated by the `&`-symbol. The regions in which the two molecules will bind to each other are underlined. The dot-bracket-notation is not able to

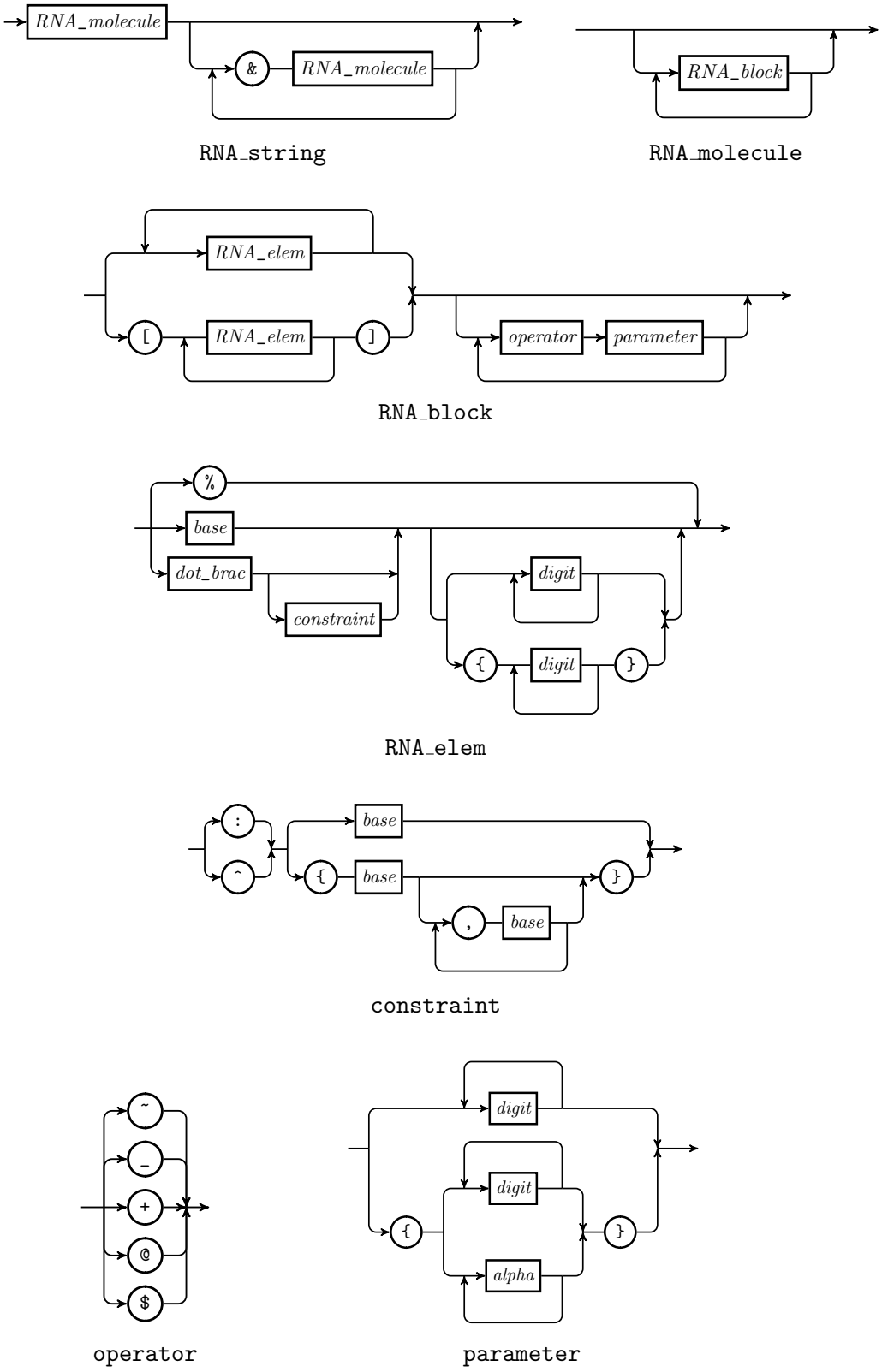


Figure 2: Syntax graph for the extended dot-bracket notation.

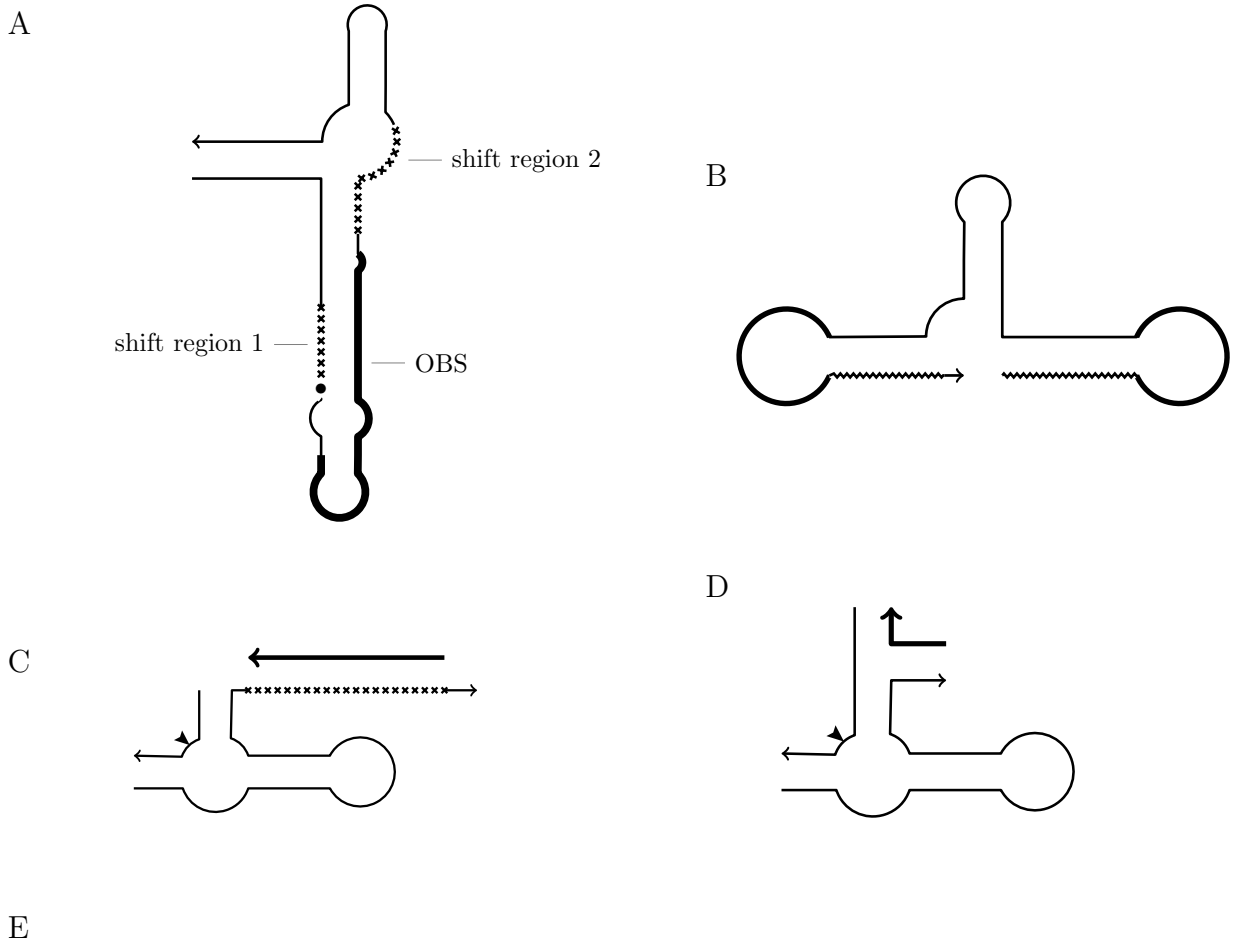
express in which combination the binding regions will bind. In larger molecules the situation can easily be more ambiguous with numerous plausible locations for intermolecular binding. In the extended dot-bracket-notation, the `+` operator can indicate the matching regions. Accordingly, the two molecules shown above can be represented as:

```
(6+{B1}.....((((((......))))))...((((....))))...(6+{B2} &
)6+{B1}........((((((....((((....))))....))))))...)6+{B2}
```

A more relevant example for such an ambiguous binding situation can be seen in Fig. 4D, where the two binding sites in the AND gate have the same length. The AND gate uses two effector molecules as input signals and, in its active state, is a three-molecule supramolecular complex. The alphanumeric marking of binding sites in the extended dot-bracket-notation enables the description of interactions among several molecules. Note in the examples above, how the standard notation and the extended notation can be mixed to highlight particular features of a molecule or set of molecules.

The benefit of the extended notation is most easily seen when it is rendered as two-dimensional structures. Figure 3 shows the rendering of several sample structures from the literature. Panel A shows a ribonucleic acid PASS gate in its inactive state [10]. Panel B depicts a deoxyribonucleic acid AND gate described in [12]. The mechanisms of Fig. 3A is based on disrupting the active conformation of a nucleic acids enzyme. Upon binding of an effector molecule to the binding site shown in bold the active conformation is restored. In the gate depicted in Fig. 3B, the binding site for the substrate of a nucleic acids enzyme is blocked by intramolecular binding highlighted with a crinkled lines. Binding of effector molecules to the two binding sites shown in bold will expose the substrate binding site. Two hammerhead ribozymes with different catalytic activation strategies described in [4] and [14] are shown in Fig. 3C and D, respectively. The hammerhead ribozymes are shown in their active conformation, each with a bound effector molecule. The effector molecules are marked with bold lines and the location of the cleavage point is marked on the substrate strand. The corresponding extended notation for the four structures is shown in panel E.

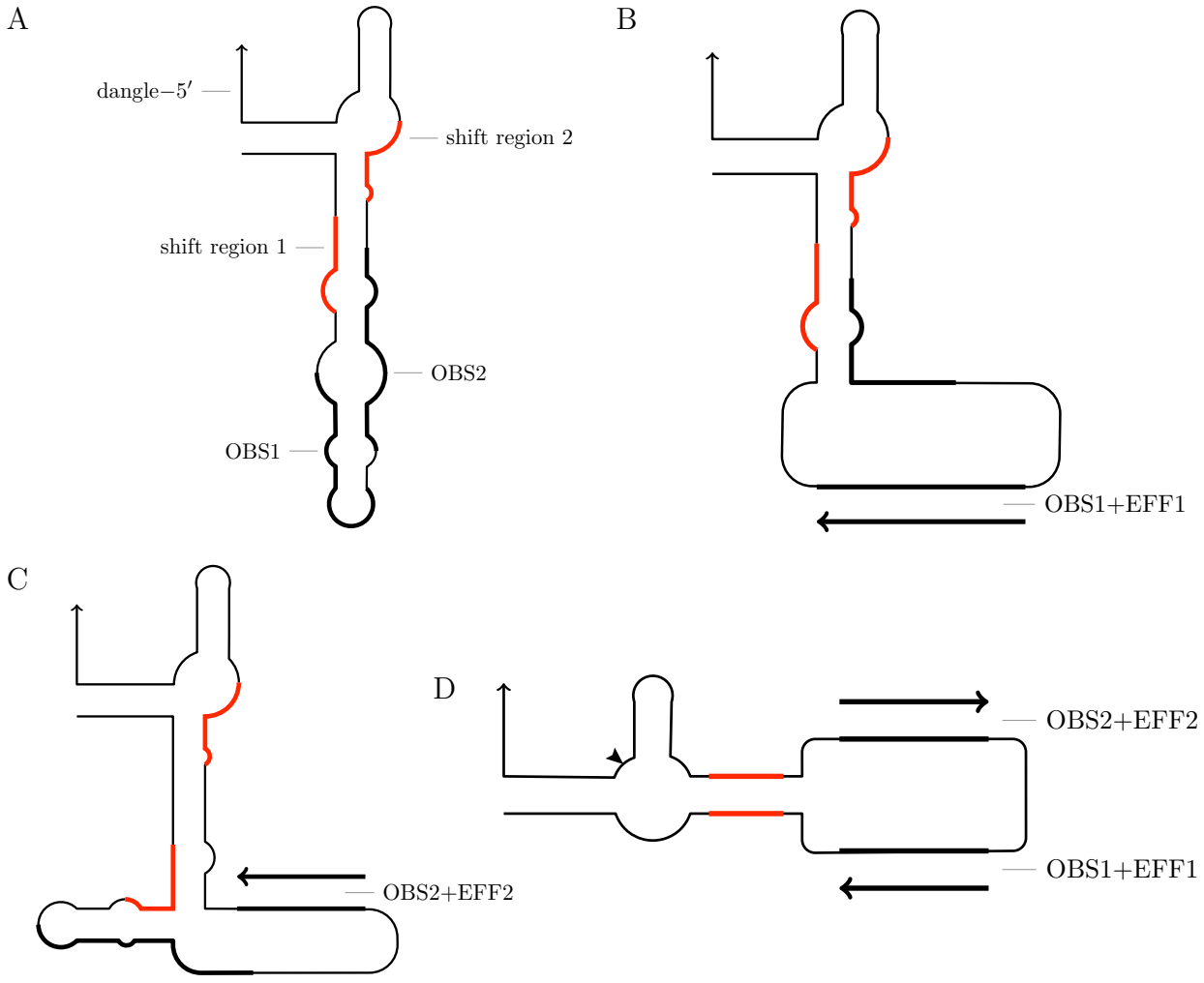
In Fig. 4 four different states of the ribonucleic AND gate designed by Penchovsky and Breaker [10] are shown. The interplay of multiple molecules and multiple conformational states is crucial to the computing schemes based on functional nucleic acids. They are also a challenge to represent in a convenient notation. Panels A shows the secondary structures of the AND gate without effector molecules, panels B and C show the structures the AND gate assumes if only one of the effector molecules is present. If both oligonucleotide binding sites (OBS1, OBS2) are occupied by effector molecules the ribozyme changes into the catalytically active conformation shown in panel D. The extended dot-bracket-notation corresponding to the four states of the gate are shown in panel E.



E

A	`({15}[(3]~2@{shift region 1}[(3]_2(.4[(2.7)3.)2]_2@{OBS}[)3.]_2 [(4.2]~2@{shift region 2}.3(5.4)5.)8`
B	`[(9]~1[.{15}]_1)9(3.4)3.5(8[.15]_1[)8]~1`
C	`[(5]+1.7(4.4)4.3[(5]+2.2[(8]+3~2.2 & [)5]+2[.]%[(5]+1 & [)8]+3_2`
D	`[(8]+1.7(4.4)4.3[(3]+2[(6]+3 & [)6]+3_1[(6]+4_1 & [)6]+4[)6]+2[.]%[)8]+1`

Figure 3: Four different structural renderings of arbitrary nucleic acid molecules.



E

A	`({15}[(4]$1@{shift region 1}[(.4]$1(3.3[.2(3.]_1@{OBS1}[(3.7]_1 )3[.)3.3]_1@{OBS2}[.3)3.2)2]_1)6[.)5]$1[.3]$1@{shift region 2}.3 (5.4)5.)8.9`
B	`({15}[(5.3]$1(3.3[({16}]+1_1.3[.9)3.2)2]_1)5[.)5.2]$1.3(5.4)5.)8.9 & [){16}]+1_1@{OBS1+EFF1}`
C	`({15}[(3.)3.]$1.2(2.3[.2)2.)3.8]_1.3[({16}]+2_1)5[.)5.2]$1.3(5.4)5.)8.9 & [){16}]+2_1@{OBS2+EFF2}`
D	`(8.7[(8]$1.6[({16}]+1_1.3[({16}]+2_1.5[)8]$1.3(5.4)5[.]%)8.9 & [){16}]+1_1@{OBS1+EFF1} & [){16}]+2_1@{OBS2+EFF2}`

Figure 4: RNA molecular AND gate after [10] in different states. Rendered from the extended dot-bracket notation.

3. Conclusions

Within recent years nucleic acids of up to about 200 nucleotides in length have become a focus of interest for prototype implementations of molecular computing concepts. During the same period the importance of ribonucleic acids as components of the regulatory networks within living cells has increasingly been revealed. While the configuration of the nucleic acids is generally linear, they can adopt a range of conformations. The conformation adopted by a nucleic acid is determined by the possibility of its nucleotide chain to form non-covalent intramolecular bonds through hybridisation of complementary regions in the nucleotide sequence. This intramolecular hybridisation pattern, known as secondary structure of the molecule, is crucial to the interactions a nucleic acid undergoes with other molecules. For a given secondary structure there is typically a large number of sequences that will adopt this structure. The standard method for denoting nucleic acid secondary structure is in the form of a string of matching parenthesis for hybridising pairs of nucleotides separated by dots representing nucleotides that do not participate in internal hybridisation. We have extended this notation for the convenience of human users as well as machine processing. The extensions allow for a more compact notation through the use of iterator operators and grouping symbols, provide for constraints placed on the nucleotides that may appear in a position, and facilitate the annotation of sequence regions and the graphical rendering of secondary structures.

A downside of the proposed annotation is the potentially increased complexity of the strings that represent a molecule. For human readers this means that more symbols and the scoping of operators needs to be understood to read the notation. For machines establishing the equivalence of the secondary structures denoted by two strings is no longer as simple as comparing the strings. However, the language of the dot-bracket notation is a subset of the extended dot-bracket notation proposed here and the use of its symbols and operators is thus wholly optional. The most appropriate notation will depend on the application. For instance, the use of iterator operators make it easier for humans to compare the length of hybridised regions within a molecule, but the relative lengths of oligonucleotides is more readily apparent if denoted without iterator operators. Human users can choose to use the features of the extended notation according to application. For machine processing the extended notation can easily be expanded or reduced to the standard dot-bracket notation.

In our own work we felt the need for a more expressive notation for computational nucleic acids. The extended dot-bracket notation described here answers this need. We are currently converting our software tools to the new notation. The accompanying widening of its use is likely to lead to further refinements of the notation and we welcome suggestions for improving its usefulness. For example, the application of iterators to groups and the nesting of groups are not currently part of the extended notation, but could be added if needed. We expect to make the code used for rendering secondary structures in LaTeX [7] with TikZ [13] available in the near future.

References

[1] ADLEMAN, L.M., Molecular computation of solutions to combinatorial problems, Science 226 (1994), 1021–1024.

[2] HOFACKER, I.L., FONTANA, W., STADLER, P.F., BONHOEFFER, L.S., TACKER, M., SCHUSTER, P., Fast folding and comparison of RNA secondary structures, Chemical Monthly 125 (2) (1994), 167–188.

[3] ANDRONESCU, M., ANGUIRRE-HERNÁNDEZ, R., CONDON, A., HOOS, H.H., RNAsoft: A suite of RNA secondary structure prediction and design software tools, Nucleic Acid Research 31 (13) (2003), 3461–3422.

[4] BURKE, D.H., OZEROVA, N.D.S., NILSEN-HAMILTON, M., Allosteric hammerhead ribozyme TRAPs, Biochemistry 41 (2002), 6588–6594.

[5] BUSCH, A., BACKOFEN, R., INFO-RNA–a fast approach to inverse RNA folding, Bioinformatics 22 (15) (2006), 1823–1831.

[6] HIGGS, P.G., RNA secondary structure: physical and computational aspects, Quarterly Reviews of Biophysics 33 (3) (2000), 199–253.

[7] LAMPORT, L., LaTex: A document preparation system, user's guide and reference manual, Addison-Wesley, Mass 1994.

[8] LIBERMAN, E.A., Analog-digital molecular cell computer, BioSystems 11 (1979), 111–124.

[9] STOJANOVIC, M.N., STEFANOVIC, D., A deoxyribozyme-based molecular automaton, Nature Biotechnology 21 (9) (2003), 1069–1074.

[10] PENCHOVSKY, R., BREAKER, R.R., Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes, Nature Biotechnology 23 (11) (2005), 1424–1433.

[11] RAMLAN, E.I., ZAUNER, K.-P., Nucleic acid enzymes: The fusion of self-assembly and conformation computing, International Journal of Unconventional Computing, In print (2008).

[12] STOJANOVIC, M.N., MITCHELL, T.E., STEFANOVIC, D., Deoxyribozyme-based logic gates, Journal of the American Chemical Society 124 (2002), 3555–3561.

[13] TANTAU, T., TikZ and PGF Manual for version 2.0, (2007).

[14] WANG, D.Y., LAI, H.Y., FELDMAN, A.R., SEN, D., A general approach for the use of oligonucleotide effectors to regulate the catalysis of RNA-cleaving ribozymes and DNAzymes, Nucleic Acids Research 30 (8) (2002), 1735–1742.

[15] ZAUNER, K.-P., Molecular information technology, Critical Reviews in Solid State and Material Sciences 30 (1) (2005), 33–69.