

rxntoarb

v. 2.30 and later

Christian Biscombe

13 November 2019

rxntoarb (pronounced reaction-to-arb) converts a human-readable system of chemical reactions into a format suitable for use with *arb* finite volume solver. Reading from one or more input files (with the preferred extension *.rxn*), **rxntoarb** automatically creates

- rate constant definitions (performing unit conversions where necessary),
- reaction rate expressions,
- source terms,
- transport equations for each chemical species (easily customisable by means of a template file),
- magnitude estimates.

Other features include easy switching between ODE and PDE output and the ability to include or exclude reactions based on regular expression matches.

rxntoarb offers several benefits:

- the reaction network is represented using intuitive syntax that is easier and faster to read and write,
- reactions may be easily added or removed without the user having to keep track of the effects of these changes on source terms,
- the potential for error is greatly reduced by eliminating the need for manual unit conversions and manual entry of reaction rates and source terms.

rxntoarb requires Ruby 1.9.3 or later. The current version of the template file (see § 2) is designed for *arb* 0.60. (Older versions of the template file are available going back to *arb* 0.56.)

Contents

1	Invocation and command-line options	3
2	File structure	4
2.1	.rxn file	4
2.2	Template file (rxntoarbc)	4
2.3	.arb output file	4
2.4	Main .arb file	4
3	.rxn file syntax	7
3.1	options statement	7
3.2	Comments	7
3.3	Headers	7
3.4	Species and region names	7
3.5	surface_region and volume_region statements	8
3.5.1	Lazy region identifiers @s and @v	8
3.6	Reactions	9
3.6.1	Irreversible reactions	9
3.6.2	Reversible reactions	9
3.6.3	Two-step reactions	9
3.6.4	Michaelis–Menten reactions	10
3.7	Kinetic parameters	10
3.7.1	Inheritance	11
3.8	Aliases	11
3.9	initial_species statement	11
3.10	Metaspecies	12
3.11	include_only and exclude statements	12
3.12	substitute statement	13
4	Template file syntax	13
4.1	Substitution sequences	13
4.2	if_rxn statement	14
4.3	each blocks	14
5	Copyright and licence	15

1 Invocation and command-line options

`rxntoarb` is invoked as follows:

```
rxntoarb [options_list] <rxn_file_list>,
```

where `rxn_file_list` specifies the path to at least one `.rxn` file and the optional `options_list` may include any of the following:

- `-d|--debug` Write copious debugging output to `STDERR`. (Intended mainly as a developer option.)
- `-i|--interactive` Prompt for confirmation before overwriting an existing `.arb` output file.
- `-l|--alias-labels` Use reaction aliases (§ 3.8) instead of internal reaction labelling scheme.
- `-m|--manual` Open this manual with the default PDF viewer. Failing that, print path to this manual.
- `-n|--none-centred` Activate ODE mode: generate ODEs (no spatial dependence) rather than PDEs. `rxntoarb` operates in PDE mode by default.
- `-o|--outfile <output_file>` Use `output_file` as the `.arb` output file. This option is ignored if `rxn_file_list` contains more than one `.rxn` file.
- `-s|--strict` Regard warnings as errors.
- `-t|--template <template_file>` Use `template_file` as the template file.
- `-v|--version` Print version information.

As of *arb* 0.58, `rxntoarb` is invoked automatically by *arb* if an `INCLUDE` statement (or similar) in an *arb* input file references a file with the `.rxn` extension. This mechanism ensures that the `.arb` output file generated by `rxntoarb` is always up-to-date with the `.rxn` file. To enable options to be passed to `rxntoarb` via this mechanism, any of the above command-line options may also be set within the `.rxn` file itself (see § 3.1).

2 File structure

Using `rxntoarb` with *arb* involves the use of four files.

2.1 .rxn file

.rxn files are input files read by `rxntoarb`. They contain a list of chemical reactions together with kinetic data. The syntax used in .rxn files is defined in § 3. An example .rxn file, reproduced in Listing 1, may be found under `examples/reactions.rxn` in the `rxntoarb` root directory. Vim syntax highlighting for .rxn files is provided by the file `rxn.vim` included in the `rxntoarb` root directory (type `:help usr_06` in Vim for instructions on how to set this up).

2.2 Template file (rxntoarbrc)

Template files store *arb* code that needs to be included for each chemical species defined in a .rxn file. The extended (non-*arb*) syntax used in template files is defined in § 4. `rxntoarb` uses the following hierarchy to determine the path of the template file:

1. If `rxntoarb` is invoked with the `-t` flag, the specified template file is used.
2. If a file with the name `rxntoarbrc` exists within the working directory from which `rxntoarb` is called, it is used as the template file.
3. The default `rxntoarbrc` located in the `rxntoarb` root directory is used otherwise.

The default `rxntoarbrc` should provide an adequate starting point in most cases. However, the user may wish to alter the (simulation-specific) boundary conditions listed beneath the comment ‘`# set boundary conditions here`’ or to define new per-species `OUTPUT` variables. In this case, the default template should be copied to the working directory and modified there.

2.3 .arb output file

The output of `rxntoarb` is an .arb output file, which contains *arb* code metaprogrammed from the contents of the .rxn file. The .arb output file will have the same basename as the .rxn file unless overridden by the `-o` flag.

2.4 Main .arb file

The .arb output file generated by `rxntoarb` contains *arb* code describing chemical species and the reactions occurring between them, but it is not sufficient to run a complete *arb* simulation. Additional code related to mesh setup, solver options, equations governing fluid flow, time-stepping, etc. should be included in a separate .arb file, referred to as the main .arb file. It is this main .arb file (and not the .arb output file) that should be run through *arb*.

The main .arb file needs to have the following characteristics:

- An `INCLUDE` statement (or similar) for the .rxn file must be present. (Alternatively the .arb output file may be included directly, but including the .rxn file is generally preferable.)
- Any variables used in the template file but not created by `rxntoarb` must be defined.
- Replacements for any flags (strings delimited by double angle brackets `<<>>`) used in the template file should be defined. The default `rxntoarbrc` contains the following flags:
 - `<<specieshighorderadvection>>`: whether to use high-order advection for chemical species; should be either 1 (true) or 0 (false; default). If false, low-order advection will be used.
 - `<<zeroinitialvolumeconcentrations>>`: whether to set species concentrations to zero initially in volume regions; should be either 1 (true) or 0 (false; default). If false, species concentrations will be set equal to their boundary values (see below) throughout volume regions.
 - `<<calculatediffusivities>>`: whether species diffusivities should be calculated using a correlation based on molecular weight; should be either 1 (true) or 0 (false; default). If true, molecular weights must be defined for all volume species (see below). If false, diffusivities must be defined for all volume species (see below).
- Species-specific physical parameters, currently only molecular weights or diffusivities, should be defined as `CONSTANTS`. Molecular weight variable names must follow the convention `<MW_speciesname>`. Diffusivity variable names must follow the convention `<D_speciesname>`. Molecular weights of complexes (two or more species bound to each other) need not be defined: they will be calculated automatically as the sum of the molecular weights of the components.
- Boundary concentrations of all species present initially should be defined as `CONSTANTS` *after* the .arb output file is included (because this file sets the boundary concentrations of all species to zero by default). For surface species, the boundary concentration corresponds to the initial concentration of that species. For volume species, the boundary concentration will generally be the concentration of that species on the boundary where it enters the volume. Boundary concentration variable names must follow the convention `<c_speciesname@region_0>` for species located in volume regions and `<s_speciesname@region_0>` for species located on surface regions.

An example main .arb file may be found under `examples/main.arb` in the `rxntoarb` root directory.

Listing 1 An example .rxn file (examples/thrombin_generation_JC_model.rxn)

```
1  ! Coagulation biochemistry from Jordan & Chaikof (2011), Biophys. J. 101: 276-286.
2  ! Reactions below are named such that (e.g.) S2.6 is Reaction 6 in Table S2.
3  ! Christian Biscombe, 2019-05-23

5  options -l -n
6  initial_species [II, V, VII, VIIa, VIII, IX, X, PC, AT, TFPI, MCA]@domain
7  initial_species TF@TF_patch, TM@TM_patch
8  exclude /@TM_patch|bA?PC\b/ # suppress reactions involving TM

10 # Fluid-phase reactions
11 volume_region domain
12 S2.6: VIII@v {IIa@v}-> VIIa@v; kcat=0.9 s-1, KM=0.20 uM
13 S2.8: V@v {IIa@v}-> Va@v; kcat=0.23 s-1, KM=0.072 uM
14 S2.9: VII@v {Xa@v}-> VIIa@v; kcat=5.0 s-1, KM=1.2 uM
15 S2.11: VII@v {IIa@v}-> VIIa@v; kcat=0.061 s-1, KM=2.7 uM
16 S3.5: IIa@v + AT@v -> AT:IIa@v; k=6.8e-3 uM-1 s-1
17 S3.6: IXa@v + AT@v -> AT:IXa@v; k=2.6e-3 uM-1 s-1
18       Xa@v + AT@v -> AT:Xa@v
19 S3.7: Xa@v + TFPI@v <=> TFPI:Xa@v; ka=16 uM-1 s-1, kd=3.3e-4 s-1

21 # Reactions on TF and TM patches
22   # Problems with units in S2.4, S2.10, S3.2, S3.4.
23   # Source terms for S2.4 and S2.10 must be manually relocated to TF and TM patches.
24 surface_regions TF_patch, TM_patch
25 S2.2: II@v {Va:Xa@s}-> IIa@v; kcat=33 s-1, KM=0.21 uM
26 S2.4: Va@v {APC@v}-> Va*@v; kcat=0.58 s-1, KM=0.025 uM
27       VIIIa@v {APC@v}-> VIIIa*@v
28 S2.7: X@v {VIIIa:IXa@s}-> Xa@v; kcat=20 s-1, KM=0.16 uM
29 S2.10: V@v {Xa@v}-> Va@v; kcat=0.046 s-1, KM=0.01 uM
30 S3.2: Va@v + Xa@v <=> Va:Xa@s; ka=100 uM-1 s-1, kd=0.01 s-1
31 S3.4: VIIIa@v + IXa@v <=> VIIIa:IXa@s; ka=100 uM-1 s-1, kd=0.01 s-1

33 # Reactions on TF patch only
34 surface_region TF_patch
35 S2.1: X@v {TF:VIIa@s}-> Xa@v; kcat=1.2 s-1, KM=0.45 uM
36 S2.5: IX@v {TF:VIIa@s}-> IXa@v; kcat=0.34 s-1, KM=0.17 uM
37 S3.1: VIIa@v + TF@s <=> TF:VIIa@s; ka=100 uM-1 s-1, kd=0.06 s-1
38 S3.8: TF:VIIa@s + TFPI:Xa@v <=> TF:VIIa:TFPI:Xa@s; ka=10 uM-1 s-1, kd=1.1e-3 s-1

40 # Reactions on TM patch only
41 surface_region TM_patch
42 S2.3: PC@v {TM:IIa@s}-> APC@v; kcat=5.58 s-1, KM=0.7 uM
43 S3.3: IIa@v + TM@s <=> TM:IIa@s; ka=100 uM-1 s-1, kd=0.01 s-1
```

3 .rxn file syntax

3.1 options statement

The options listed in § 1 may also be set within a .rxn file using an **options** statement, which (if used) should appear near the top of the file. The syntax for setting options exactly mirrors the command-line invocation. For example, line 5 in Listing 1 changes the internal reaction labelling scheme (for convenience) and activates ODE mode. Note that options set using an **options** statement take precedence over options set on the command line.

3.2 Comments

Comments begin with the **#** character and continue until the end of the line. Comments may appear anywhere in the .rxn file. Full-line comments (e.g. line 10) are ignored by **rxntoarb**. Comments that appear after the parameter list of a reaction will be carried over into the .arb output file, where they appear after the **CONSTANT** expressions that define the relevant kinetic parameters.

3.3 Headers

Lines beginning with a bang (!) are called headers (e.g. lines 1–3). Headers are a special type of comment designed primarily to store metadata about the reaction network (title, author, date, description etc.). As the name suggests, it is intended that headers appear only at the top of the .rxn file. Headers will be collected together as comments at the top of the .arb output file. A line showing the date and time at which the .arb output file was generated and the version of **rxntoarb** used will be appended to the header automatically.

3.4 Species and region names

Chemical species have the general form **speciesname@region**. The **speciesname** describes the chemical identity of the species; the **@region** specifies the arb region on/in which that species is present. (Note that **@** is not part of the region name, but rather acts to separate the species and region names.) Chemical species are declared by including them in a reaction (see § 3.6).

Species and region names may not contain any of the characters **<>'"#&** or the character sequences **{** or **}** (restrictions inherited from *arb*). **rxntoarb** also requires that species and region names not contain any of the characters **,;@** or any of the substitution sequences or extended syntax used in template files (see § 4). Colons (:) within species names are intended to separate the components of a complex (e.g. **AT:IIa** in line 16 is a complex of **AT** and **IIa**), but this interpretation only comes into play in calculating the molecular weight of complexes. The use of *arb*-style angle bracket delimiters (**<>**) is necessary for species names only if they begin with digits (which would otherwise be

interpreted as a stoichiometric coefficient), and necessary for region names only if they contain characters other than letters, digits, and underscores.

Two lazy region identifiers, `@s` and `@v`, are predefined (see § 3.5.1). These identifiers interact with `surface_region` and `volume_region` statements. As such, `s` and `v` should not be used as user-defined region names.

When `rxntoarb` is called with the `-n` flag, ODE mode is activated. In this mode, all species and reactions are assumed to have no spatial dependence and hence region names effectively become part of the species name.

It is permissible to omit `@region` identifiers throughout the `.rxn` file under two circumstances:

- No `surface_region` or `volume_region` statements appear. In this case all species and reactions will be assumed to have no spatial dependence, i.e. ODE mode will be assumed.
- Exactly one surface or volume region is specified by either a `surface_region` or `volume_region` statement. In this case all species and reactions will be assumed to be located on/in that region.

3.5 `surface_region` and `volume_region` statements

The `surface_region` and `volume_region` keywords (both optionally plural) are used to specify *arb* regions on/in which chemical species are present and reactions occur. The keywords should be followed by a comma-separated list of region names (conforming to the naming rules stipulated in § 3.4). If `rxntoarb` is called without the `-n` flag (PDE mode), surface regions will be assumed to be two-dimensional and volume regions will be assumed to be three-dimensional. If `rxntoarb` is called with the `-n` flag (ODE mode), then the dimensionality of regions is ignored.

`surface_region` and `volume_region` statements apply to all reactions subsequently defined in the `.rxn` file, unless and until overridden by another `surface_region` or `volume_region` statement.

3.5.1 Lazy region identifiers `@s` and `@v`

The lazy region identifiers `@s` and `@v` are shorthand references to the surface and volume regions specified by the previous `surface_region` and `volume_region` statements. For example, the `surface_regions` statement in line 24 specifies two surface regions, `TF_patch` and `TM_patch`. References to `@s` in lines 25–31 will therefore be interpreted by `rxntoarb` as references to both `TF_patch` and `TM_patch`, whilst references to `@v` will be interpreted as references to the volume region domain specified by the `volume_region` statement in line 11. In line 34, the new `surface_region` statement overrides the previous one so that references to `@s` in lines 35–38 will be interpreted as references to `TF_patch` only.

3.6 Reactions

Four types of reactions are recognised, as described below. Reactions may be entered in any order in the .rxn file, subject to the condition that the lazy region identifiers `@s` and `@v` will take their value(s) from the previous `surface_region` and `volume_region` statements. With the exception of Michaelis–Menten reactions (whose kinetics are governed by the [Michaelis–Menten equation](#)), all reaction rates are assumed to be described by [mass action kinetics](#).

3.6.1 Irreversible reactions

Irreversible reactions take the form

```
reactant_list -> product_list.
```

`reactant_list` consists of one or more species, separated by spaces and the character `+` (the spaces are mandatory so that `+` can also be used in species names, e.g. for ions). Stoichiometric coefficients (if any) should immediately precede the species name, optionally separated from it by a space or the `.` or `*` characters. Non-integer stoichiometric coefficients are not supported. `product_list` is similar except that it may be empty, in which case the products of the reaction are discarded (not tracked in the *arb* simulation). For readability, spaces around the reaction arrow `->` are recommended (but not mandatory). The kinetic parameter `k` describing the rate of the reaction must either be specified or inherited (see § 3.7). Examples of irreversible reactions are shown in lines 16–18.

3.6.2 Reversible reactions

Reversible reactions take the form

```
reactant_list <=> product_list.
```

`product_list` must not be empty for a reversible reaction. The kinetic parameters `ka` and `kd` describing the rates of the forward and reverse reactions must either be specified or inherited (see § 3.7). Examples of reversible reactions are shown in lines 30–31.

3.6.3 Two-step reactions

Two-step reactions consist of a reversible reaction followed by an irreversible reaction:

```
reactant_list <=> intermediate_list -> product_list.
```

`intermediate_list` must not be empty but `product_list` may be empty (which implies that the products are not tracked in the *arb* simulation). A shorthand syntax is available for the case in which the only intermediate is a complex formed from all of the reactants (this intermediate will be named automatically based on the reactant names):

```
reactant_list => product_list.
```

In both full and shorthand forms, the kinetic parameters `ka`, `kd`, and `k` describing the rates of the forward and reverse reactions in the first step and the irreversible reaction in the second step must either be specified or inherited (see § 3.7).

3.6.4 Michaelis–Menten reactions

Michaelis–Menten reactions take the form

```
substrate {enzyme}-> product_list.
```

Exactly one `substrate` and one `enzyme` must be specified. `product_list` may be empty (which implies that the products are not tracked in the *arb* simulation). The Michaelis–Menten parameters `KM` and `kcat` must either be specified or inherited (see § 3.7). Examples of Michaelis–Menten reactions are shown in lines 12–15.

3.7 Kinetic parameters

Kinetic parameters are entered as a comma-separated list on the same line as the reaction they describe, separated from the reaction itself by a semicolon (;). Kinetic parameters may be entered in any order. The following parameter names are recognised:

- `k`,
- `ka` (synonyms `kf` and `kon`),
- `kd` (synonyms `kr` and `koff`),
- `KM` (synonym `Km`),
- `kcat`.

These names may optionally be followed by a `*` character, which causes the following numerical value to be output as an expression (enclosed in quotes) rather than a numerical constant (*arb* handles expressions and numerical constants differently, which can be important for redefinitions).

Kinetic parameter values may be entered in two forms. In the first form (used throughout the example `.rxn` file), a numerical value together with appropriate physical units are specified. The numerical value and each individual unit should be separated by spaces. SI prefixes immediately precede the unit name. Exponents on units should immediately follow the unit name, optionally preceded by the `^` character. More information on unit entry is given in the accompanying documentation for the `convert_units` program, which provides a command-line interface to the `Units` module that accompanies `rxntoarb`. Where necessary, `rxntoarb` performs unit conversions so that parameter values in the `.arb` output file are expressed in SI base units. Unit consistency checking is also performed with warning messages issued if reaction rates have inconsistent or unexpected units. (Such warnings are produced by lines 30–31.)

Alternatively, kinetic parameter values may be specified as strings, delimited by either single or double quotes. Strings may contain arbitrary *arb* code. This feature is useful in conjunction with aliases (see § 3.8) or for expressing one kinetic parameter in terms of others. The downside is that unit consistency checking is not possible.

3.7.1 Inheritance

Sometimes the same kinetic parameters will apply to a group of closely related reactions. Kinetic parameters from one ‘parent’ reaction can be assigned to any number of ‘child’ reactions by placing the children directly below the parent and indenting them by at least one space. An example of inheritance is shown in lines 17–18: the reaction in line 18 inherits the kinetic parameters from line 17. The advantage of inheritance is that kinetic parameters for the group of reactions need only be entered in one place.

3.8 Aliases

Any non-indented reaction may optionally be preceded by an alias, which runs from the beginning of the line until a colon (:) followed by a (mandatory) space is encountered. Aliases serve two purposes. Firstly, they may be used to provide descriptive names for reactions. If `rxntoarb` is invoked with the `-l` flag (as the example `.rxn` file is), these names will be used throughout the `.arb` output file in place of the automatic labels (derived from species names) otherwise generated by `rxntoarb`. Secondly, aliases provide a convenient way to refer to kinetic parameters defined elsewhere in the `.rxn` file. For example, lines 17–18 in Listing 1 could alternatively be written

```
17 S3.6: IXa@v + AT@v -> AT:IXa@v; k=2.6e-3 uM-1 s-1
18 Xa@v + AT@v -> AT:Xa@v; k="<k_S3.6>"
```

In this case, the value of `k` specified in line 17 (alias `S3.6`) will also be assigned to the reaction in line 18.

3.9 initial_species statement

An `initial_species` statement is used to declare which species are present initially in the *arb* simulation. This information is used by `rxntoarb` to estimate the magnitudes of the concentrations of all species based on the concentrations of the precursor species from which they are derived.

The `initial_species` keyword should be followed by a comma-separated list of species together with their associated regions, as shown in line 7. If multiple species on/in the same region are present initially, the array notation shown in line 6 may be used. Any species appearing in an `initial_species` statement but otherwise not appearing in the `.rxn` file will be ignored.

See § 2.4 for details on how boundary concentrations must be specified in the main `.arb` file.

3.10 Metaspecies

A ‘metaspecies’ is a group of entities that behaves like a single entity in a reaction. For instance, when coagulation factors X and Xa bind to phospholipid membranes composed of phosphatidylcholine and phosphatidylserine (70:30 mixture), they occupy 106 and 52 phospholipid head groups, respectively.¹ These binding reactions should *not* be written as

```
X@v + 106PL@s <=> X@s; ka=..., kd=... # INCORRECT!
Xa@v + 52PL@s <=> Xa@s; ka=..., kd=... # INCORRECT!
```

because under mass action kinetics, the rates of the reactions as written above would be proportional to the concentration of PL@s raised to the power of 106 and 52, respectively, which is not what is wanted. Instead, binding sites should be entered using the following square bracket notation to represent a metaspecies:

```
X@v + [106PL@s] <=> X@s; ka=..., kd=...
Xa@v + [52PL@s] <=> Xa@s; ka=..., kd=...
```

`rxntoarb` interprets the metaspecies as single entities so that the rates of these reactions are proportional to the concentrations of the respective binding sites (equal to the concentration of PL@s divided by 106 and 52, respectively). The source term for PL@s is handled sensibly too.

As metaspecies behave like any other chemical species, they may be preceded by a stoichiometric coefficient if necessary (e.g. `2[106PL@s]`).

3.11 `include_only` and `exclude` statements

`include_only` and `exclude` are used to selectively include or exclude regions or reactions in the `.arb` output file. Both keywords must be followed either by (i) a whitespace-free text string for simple text matching or (ii) a valid Ruby regular expression for more complex matches. (Regular expressions are useful for specifying alternate match patterns, forcing the statement to respect word boundaries, or preventing end-of-line comments from being scanned. The Regexp option `i`, which enables case-insensitive matching, is the only Regexp option recognised by `rxntoarb`.) For the statement to operate on a region, the region name must be preceded by the `@` character. For example, the `exclude` statement at line 8 excludes all reactions occurring on the `TM_patch` region as well as all fluid-phase reactions involving the species `PC` and `APC`.

`include_only` and `exclude` statements apply to all subsequent lines in the `.rxn` file. Multiple `include_only` or `exclude` statements may be given, in which case reactions will be tested against all inclusion/exclusion criteria.² End-of-line comments are scanned by default, effectively enabling the user to define their own keywords to select/deselect subsets of reactions. To prevent end-of-line comments from being scanned, use a regular expression beginning with `^[^#]*`.

¹ Hathcock et al. (2005), *Biochemistry* 44: 8187.

² Before version 2.29, multiple `include_only` or `exclude` statements were not permitted.

3.12 substitute statement

A `substitute` statement³ performs code replacements. It is particularly useful when metaspecies appear in more than one reaction (although its use is not limited to this application). Carrying on the example from § 3.10, suppose it is decided that factors X and Xa should both bind to 79 phospholipid head groups (79 is the average of 106 and 52). To avoid repeating the constant 79 in multiple locations, a pointer `S@s` to the binding site metaspecies is defined via a `substitute` statement as follows:

```
substitute /\bS@s\b/ [79PL@s]  
X@v + S@s <=> X@s; ka=..., kd=...  
Xa@v + S@s <=> Xa@s; ka=..., kd=...
```

When the above code is parsed, all instances of `S@s` below the `substitute` statement will be replaced internally by `[79PL@s]`. (Including `\b` in the regular expression ensures that only complete matches are replaced; other species that happen to contain `S@s`—e.g. `PS@s`—will not be affected.)

In general, the `substitute` keyword takes two space-separated arguments. The first argument must either be (i) a whitespace-free text string for simple text matching or (ii) a valid Ruby regular expression for more complex matches. (The Regexp option `i`, which enables case-insensitive matching, is the only Regexp option recognised by `rxntoarb`.) The second argument is taken to be any text following the first argument (excluding end-of-line comments starting with the `#` character). All subsequent instances of text matching the first argument will be replaced with the second argument.

In principle, `substitute` statements could be used to override any aspect of the syntax described in this manual. Judiciousness is strongly advised.

4 Template file syntax

`rxntoarb` template files (called `rxntoarbrc` by default) store *arb* code that needs to be included for each species. A range of substitutions are performed on the original template file to generate unique code for each species. Template files contain extended (non-*arb*) syntax that is handled by `rxntoarb` as detailed below.

4.1 Substitution sequences

`rxntoarb` performs substitutions on the following character sequences (all delimited by forward slashes), which must therefore not appear in any species or region names:

- `/c/` (concentration of the species),
- `/species/` (species name),
- `/region/` (region name),

³ The `substitute` statement extends and replaces the `let` statement that appeared in versions 2.7–2.17.

- `/source_region/` (region name),
- `/CENTRING/` (NONE|FACE|CELL depending on species location),
- `/centring/` (none|face|cell depending on species location),
- `/units/` (''|mol m-2|mol m-3 depending on species location),
- `/MW/` (molecular weight of the species),
- `/associatedfaces(region)/` (region consisting of faces associated with the region),
- `/associatedcells(region)/` (region consisting of cells associated with the region),
- `/domainof(region)/` (region consisting of cells in domain of the region).

Additionally, all *arb* statements of the form `ON <region>` are deleted in ODE mode.

4.2 `if_rxn` statement

`if_rxn` statements are used to conditionally include *arb* code depending on whether a species is located on a surface, in a volume, or has no location (ODE mode). `if_rxn` statements take the form

```
if_rxn(location=region_list){if_clause}{else_clause}.
```

The term in parentheses determines the scope of the `if_rxn` statement. `location` must be one of `none`, `surface`, or `volume`. The optional `region_list` limits the scope of the `if_rxn` statement to only the named regions, and should be a comma-separated list of region names (*arb*-style angle bracket delimiters are optional). The code contained in the mandatory `if_clause` is retained for all species defined within the scope of the `if_rxn` statement and deleted otherwise. The code contained in the optional `else_clause` is retained for all species outside the scope of the `if_rxn` statement and deleted otherwise. Both `if_clause` and `else_clause` may span multiple lines. Nested brace pairs within clauses are allowed but nested `if_rxn` statements are not permitted.

4.3 `each` blocks

`each` blocks are used to apply a particular boundary condition to multiple surface regions bounding a volume region, and hence must reside within `if_rxn(volume)` statements. `each` blocks take the Ruby-inspired form

```
[region_list].each { |loopvar| expression }.
```

The mandatory `region_list`, which specifies the volume-region-bounding surface regions for which the `expression` in the block will be retained, should be a comma-separated

list of region names (*arb*-style angle bracket delimiters are optional). Each of these region names (stripped of angle brackets, if any) will be passed to the block, where they are accessed through the (user-named) variable `|loopvar|` (vertical bars mandatory). In `region_list`, the shorthand `*` may be used to represent all volume-region-bounding surface regions on which reactions are occurring.

`each` blocks may span multiple lines. Nested brace pairs within `each` blocks are allowed but nested `each` blocks are not permitted.

5 Copyright and licence

`rxntoarb` source code and documentation © 2016–2019 Christian Biscombe.

`rxntoarb` is contributed to *arb* finite volume solver (in which copyright is held by Dalton Harvie) under the same licence terms as that project. At the time of writing, *arb* is released under the terms of the GNU General Public License (version 3) as published by the Free Software Foundation.