

# CAS 741: Module Interface Specification

## Aqueous Speciation Diagram Generator

Steven Palmer  
`palmes4`

December 18, 2017

# Revision History

Table 1: Revision History

Date	Developer(s)	Change
11.29.2017	S. Palmer	Revision 0 submission
12.18.2017	S. Palmer	Revision 1

# Symbols, Abbreviations and Acronyms

See [SRS Documentation](#).

# Contents

<b>Revision History</b>	<b>i</b>
<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Notation</b>	<b>1</b>
<b>3 Module Decomposition</b>	<b>2</b>
<b>4 MIS of External Interface</b>	<b>4</b>
4.1 Module . . . . .	4
4.2 Uses . . . . .	4
4.3 Syntax . . . . .	4
4.3.1 Re-Exported Types . . . . .	4
4.3.2 Re-Exported Access Programs . . . . .	4
4.4 Semantics . . . . .	4
<b>5 MIS of Chemical System</b>	<b>5</b>
5.1 Template Module . . . . .	5
5.2 Uses . . . . .	5
5.3 Syntax . . . . .	5
5.3.1 Exported Types . . . . .	5
5.3.2 Exported Access Programs . . . . .	5
5.4 Semantics . . . . .	5
5.4.1 State Variables . . . . .	5
5.4.2 Environment Variables . . . . .	5
5.4.3 Access Routine Semantics . . . . .	5
<b>6 MIS of Species</b>	<b>8</b>
6.1 Template Module . . . . .	8
6.2 Uses . . . . .	8
6.3 Syntax . . . . .	8
6.3.1 Exported Types . . . . .	8
6.3.2 Exported Access Programs . . . . .	8
6.4 Semantics . . . . .	8
6.4.1 State Variables . . . . .	8
6.4.2 Access Routine Semantics . . . . .	8
<b>7 MIS of Reaction</b>	<b>10</b>
7.1 Template Module . . . . .	10
7.2 Uses . . . . .	10

7.3	Syntax . . . . .	10
7.3.1	Exported Types . . . . .	10
7.3.2	Exported Access Programs . . . . .	10
7.4	Semantics . . . . .	10
7.4.1	State Variables . . . . .	10
7.4.2	Access Routine Semantics . . . . .	10
<b>8</b>	<b>MIS of Input Conversion</b>	<b>12</b>
8.1	Module . . . . .	12
8.2	Uses . . . . .	12
8.3	Syntax . . . . .	12
8.3.1	Exported Access Programs . . . . .	12
8.4	Semantics . . . . .	12
8.4.1	State Variables . . . . .	12
8.4.2	Access Routine Semantics . . . . .	12
<b>9</b>	<b>MIS of Calculation</b>	<b>14</b>
9.1	Module . . . . .	14
9.2	Uses . . . . .	14
9.3	Syntax . . . . .	14
9.3.1	Exported Access Programs . . . . .	14
9.4	Semantics . . . . .	14
9.4.1	State Variables . . . . .	14
9.4.2	Access Routine Semantics . . . . .	14
9.4.3	Local Functions . . . . .	14
<b>10</b>	<b>MIS of Non-Linear Solver</b>	<b>15</b>
10.1	Module . . . . .	15
10.2	Uses . . . . .	15
10.3	Syntax . . . . .	15
10.3.1	Exported Access Programs . . . . .	15
10.4	Semantics . . . . .	15
10.4.1	State Variables . . . . .	15
10.4.2	Access Routine Semantics . . . . .	15
<b>11</b>	<b>MIS of Plotting</b>	<b>16</b>
11.1	Module . . . . .	16
11.2	Uses . . . . .	16
11.3	Syntax . . . . .	16
11.3.1	Exported Access Programs . . . . .	16
11.4	Semantics . . . . .	16
11.4.1	State Variables . . . . .	16
11.4.2	Environment Variables . . . . .	16

11.4.3 Access Routine Semantics . . . . .	16
---	----

# 1 Introduction

The following document details the Module Interface Specifications for the Aqueous Speciation Diagram Generator program. Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found [here](#).

## 2 Notation

The structure of the MIS for modules comes from [Hoffman and Strooper \(1995\)](#), with the addition that template modules have been adapted from [Ghezzi et al. \(2003\)](#). All template modules described in the MIS are assumed to have getters for state variables; they will not be explicitly specified. The mathematical notation comes from Chapter 3 of [Hoffman and Strooper \(1995\)](#), as well as [Gries and Schneider \(1993\)](#). For instance, the symbol  $:=$  is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

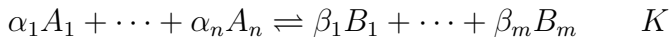
The following table summarizes the primitive data types used by SpecGen.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

The specification of SpecGen uses some derived data types: sets, sequences, strings, and tuples. Sets are an unordered unique collection of elements of the same data type. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SpecGen uses functions (partial and total), which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

Since chemical reactions, species and elements are central to SpecGen, the following notation is introduced in an attempt to simplify the presentation of the MIS.

Chemical reactions:



where  $A_i$  are reactant species,  $\alpha_i$  are the stoichiometric numbers of the reactant species,  $n$  is the number of reactant species,  $B_i$  are product species,  $\beta_i$  are the stoichiometric numbers of the product species,  $m$  is the number of product species, and  $K$  is the equilibrium constant.

Chemical species:

$$E_{e1}^1 \cdots E_{en(s)}^{n\ z}$$

where  $E^i$  are chemical elements,  $e_i$  are the number of each element  $E^i$  in the species,  $n$  is the number of distinct elements in the species,  $z$  is the charge, and  $s$  is the state. The state of a species ranges over the set of values { aqueous (aq), solid (s), liquid (l), gas (g) }.

The following pseudo-types are used in this document. The species and reaction pseudo-types are treated as strings with fields that correspond to the variables in the notations given above. Note that these types are only used for convenience with respect to specification and will not occur in the SpecGen implementation.

Pseudo Type	Notation	Resolves To
species	$\mathbb{S}$	string representation of a chemical species
reaction	$\mathbb{X}$	string representation of a chemical reaction
element	$\mathbb{E}$	string representation of a chemical element

### 3 Module Decomposition

The following table is taken directly from the Module Guide document for this project.



Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	External Interface Chemical System Species Reaction Input Conversion Calculation
Software Decision	Non-linear Solver Plotting

Table 2: Module Hierarchy

## 4 MIS of External Interface

### 4.1 Module

SpecGen

### 4.2 Uses

ChemSys (Section 5)

### 4.3 Syntax

#### 4.3.1 Re-Exported Types

ChemSys

#### 4.3.2 Re-Exported Access Programs

ChemSys  
registerRxn  
registerTotal  
specGen

### 4.4 Semantics

N/A

## 5 MIS of Chemical System

### 5.1 Template Module

ChemSys

### 5.2 Uses

Species (Section 6)

ChemEq (Section 7)

Calculation (Section 9)

Plotting (Section 11)

### 5.3 Syntax

#### 5.3.1 Exported Types

ChemSys = ?

#### 5.3.2 Exported Access Programs

Name	In	Out	Exceptions
ChemSys	-	ChemSys	-
registerRxn	$\mathbb{X}$	-	-
registerTotal	$\mathbb{E}, \mathbb{R}$	-	NonPositiveConc
specGen	fileName: String	-	-

### 5.4 Semantics

#### 5.4.1 State Variables

rxns: Set of ChemEq

species:  $\mathbb{S} \rightarrow \text{Species}$

totals:  $\mathbb{E} \rightarrow \mathbb{R}$

#### 5.4.2 Environment Variables

FileSystem: The file system of the machine running SpecGen

#### 5.4.3 Access Routine Semantics

ChemSys():

- transition:

rxns :=  $\emptyset$

```

species := {
  (H(aq)+, Species(H(aq)+, aq, 1, {(H, 1)})),
  (OH(aq)-, Species(OH(aq)-, aq, -1, {(H, 1), (O, 1)})),
  (H2O(l), Species(H2O(l), 1, 0, {(H, 2), (O, 1)})),
}
totals := ∅

```

- output:

```
out := self
```

- exception: N/A

registerRxn(rx):

- transition:

```

rxns := rxns ∪ ChemEq(
  log rx.K,
  { ∀ i : ℕ | i ≤ rx.n • (rx.Ai, rx.αi) },
  { ∀ i : ℕ | i ≤ rx.m • (rx.Bi, rx.βi) }
)
species := species ∪ { ∀ sp : S | sp ∉ dom(species) ∧ sp ∈ rx •
  (sp, Species(sp, sp.s, sp.z, { ∀ i : ℕ | i ≤ sp.n • (sp.Ei, sp.ei) } ))
}

```

- output: N/A

- exception: N/A

registerTotal(elt, tot):

- transition:

```
totals := totals ∪ { (elt, tot) }
```

- output: N/A

- exception:

```
exc := (tot ≤ 0 ⇒ NonPositiveConc)
```

specGen(fileName):

- transition:

```
FileSystem := plot(calcSpec(self), fileName)
```

- output: N/A
- exception: N/A

## 6 MIS of Species

### 6.1 Template Module

Species

### 6.2 Uses

N/A

### 6.3 Syntax

#### 6.3.1 Exported Types

Species = ?

State = {s, l, g, aq}

#### 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
Species	formula: $\mathbb{S}$ state: State charge: $\mathbb{Z}$ composition: $\mathbb{E} \rightarrow \mathbb{N}$	Species	-

### 6.4 Semantics

#### 6.4.1 State Variables

formula:  $\mathbb{S}$

state: State

charge:  $\mathbb{Z}$

composition:  $\mathbb{E} \rightarrow \mathbb{N}$

#### 6.4.2 Access Routine Semantics

Species(formula, state, charge, composition):

- transition:

self.formula := formula

self.state := state

self.charge := charge

self.composition := composition

- output:

*out* := self

- exception: N/A

## 7 MIS of Reaction

### 7.1 Template Module

ChemEq

### 7.2 Uses

N/A

### 7.3 Syntax

#### 7.3.1 Exported Types

ChemEq = ?

#### 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
ChemEq	reaction: $\mathbb{X}$ logK: $\mathbb{R}$ r: $\mathbb{S} \rightarrow \mathbb{N}$ p: $\mathbb{S} \rightarrow \mathbb{N}$	ChemEq	-

### 7.4 Semantics

#### 7.4.1 State Variables

reaction:  $\mathbb{X}$

logK:  $\mathbb{R}$

r:  $\mathbb{S} \rightarrow \mathbb{N}$

p:  $\mathbb{S} \rightarrow \mathbb{N}$

#### 7.4.2 Access Routine Semantics

ChemEq(reaction, logK, r, p):

- transition:

self.reaction := reaction

self.logK := logK

self.r := r

self.p := p

- output:



$out := self$

- exception: N/A

## 8 MIS of Input Conversion

### 8.1 Module

Convert

### 8.2 Uses

ChemSys (Section 5)

Species (Section 6)

ChemEq (Section 7)

### 8.3 Syntax

#### 8.3.1 Exported Access Programs

Name	In	Out	Exceptions
makeRootFunc	syst: ChemSys	Set of $\mathbb{R}^n \rightarrow \mathbb{R}$	-

### 8.4 Semantics

#### 8.4.1 State Variables

N/A

#### 8.4.2 Access Routine Semantics

makeRootFunc(syst):

- transition: N/A
- output:

$$\begin{aligned}
 out := & \{ \forall r : \text{ChemEq} \mid r \in \text{syst.rxn} \bullet \\
 & \lambda(\forall s : \mathbb{S} \mid s \in \text{dom}(\text{syst.species}) \wedge s.s = \text{aq} \bullet C_s) \cdot \\
 & \quad \sum(\forall s : \mathbb{S} \mid s \in \text{dom}(r.p) \wedge s.s = \text{aq} \bullet r.p(s) \cdot \log C_s) \\
 & \quad - \sum(\forall s : \mathbb{S} \mid s \in \text{dom}(r.r) \wedge s.s = \text{aq} \bullet r.r(s) \cdot \log C_s) \\
 & \quad - r.\log K \\
 & \} \cup \{ \forall e : \mathbb{E} \mid e \in \text{dom}(\text{syst.totals}) \bullet \\
 & \quad \lambda(\forall s : \mathbb{S} \mid s \in \text{syst.species} \wedge s.s = \text{aq} \bullet C_s) \cdot \\
 & \quad \log \sum(\forall s : \mathbb{S} \mid s \in \text{dom}(\text{syst.species}) \\
 & \quad \quad \wedge e \in \text{dom}(\text{syst.species}(s).\text{composition}) \bullet \\
 & \quad \quad \text{syst.species}(s).\text{composition}(e) \cdot C_s) \\
 & \quad - \log \text{syst.totals}(e) \\
 & \}
 \end{aligned}$$

- exception: N/A

## 9 MIS of Calculation

### 9.1 Module

Calculation

### 9.2 Uses

ChemSys (Section 5)

Convert (Section 8)

Solver (Section 10)

### 9.3 Syntax

#### 9.3.1 Exported Access Programs

Name	In	Out	Exceptions
calcSpec	syst: ChemSys	Set of $(\mathbb{R}, \mathbb{R}^n)$	-

### 9.4 Semantics

#### 9.4.1 State Variables

N/A

#### 9.4.2 Access Routine Semantics

calcSpec(syst):

- transition: N/A

- output:

$$\begin{aligned} out := \{ \forall pH : \mathbb{R} \mid 0 \leq pH \leq 14 \bullet \\ (pH, deLog(rootSolve(partialApply(makeRootFunc(syst), pH)))) \\ \} \end{aligned}$$

- exception: N/A

#### 9.4.3 Local Functions

partialApply:  $(\text{Set of } \mathbb{R}^n \rightarrow \mathbb{R}) \times \mathbb{R} \rightarrow (\text{Set of } \mathbb{R}^{n-2} \rightarrow \mathbb{R})$

partialApply(eqs, pH)  $\equiv \{ \forall f : eqs \bullet f(C_{H^+} = 10^{-pH}, C_{OH^-} = 10^{pH-14}) \}$

deLog:  $\mathbb{R}^n \rightarrow \mathbb{R}^n$

deLog(concs)  $\equiv \text{map}(\lambda x. 10^x, \text{concs})$

## 10 MIS of Non-Linear Solver

### 10.1 Module

Solver

### 10.2 Uses

N/A

### 10.3 Syntax

#### 10.3.1 Exported Access Programs

Name	In	Out	Exceptions
rootSolve	eqs: Set of $\mathbb{R}^n \rightarrow \mathbb{R}$ for any $n : \mathbb{N}$	$\mathbb{R}^n$	-

### 10.4 Semantics

#### 10.4.1 State Variables

N/A

#### 10.4.2 Access Routine Semantics

rootSolve(eqs):

- transition: N/A
- output:  
 $out := res, \text{ where } (\forall f : eqs \bullet f(res) = 0)$
- exception: N/A

## 11 MIS of Plotting

### 11.1 Module

Plotting

### 11.2 Uses

N/A

### 11.3 Syntax

#### 11.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	data: Set of $(\mathbb{R}, \mathbb{R}^n)$ for any $n : \mathbb{N}$ fileName: String	-	-

### 11.4 Semantics

#### 11.4.1 State Variables

N/A

#### 11.4.2 Environment Variables

FileSystem: The file system of the machine running SpecGen

#### 11.4.3 Access Routine Semantics

plot(data, fileName):

- transition: Creates a new image file with path *fileName* in *FileSystem* with a 2D plot of the set of data in *data*. The first item in each tuple of *data* is the x-value and the second item contains y-values that should be paired with that x-value.
- output: N/A
- exception: N/A

## References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*. Springer-Verlag New York, Inc., New York, NY, USA, 1993. ISBN 0-387-94115-0.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.