Module Interface Specification for Stoichiometry Mass-Mass Program

Deemah Alomair

December 6, 2019

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	First version of document

2 Symbols, Abbreviations and Acronyms

See SRS Documentation here

Contents

3 Introduction

The following document details the Module Interface Specifications for Stoichiometry Mass-Mass Program

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found here

4 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). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

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

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

The specification of SMMP uses some derived data types: sequences, strings, and tuples. 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, SMMP uses functions, 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 reaction is central to SMMP, the following notation is introduced in an attempt to simplify the presentation of the MIS.

Chemical reaction:

$$c_1 R[0]_1 + c_2 R[0]_2 = cc_1 R[1]_1 + cc_2 R[1]_2$$

where $R[0]_1$ is reactant with known mass, c_1 is the coefficient number associated to this reactant , $R[0]_2$ is reactant with unknown mass, c_2 is the coefficient number associated to this reactant , $R[1]_i$ is a product and cc_i is he coefficient number associated to this product. [The name cc_i is confusing. It looks like c times c_i. Maybe you could use a name like d_i? —SS]

5 Module Decomposition

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

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Module Atomic Mass Module ReactionT Module Mass Calculation Module
Software Decision	GUI Module

Table 1: Module Hierarchy

6 MIS of Input Module

6.1 Module

Input

6.2 Uses

Not Applicable.

6.3 Syntax

6.3.1 Exported Constants

None.

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
$input_1$	Chemical-Reaction: ReactionT	-	MismatchedInput
$input_2$	Mass-Value : $\mathbb{R}$	-	NegativeMassException

[The input module shouldn't take the input as a ReactionT. The input should build the ReactionT from the user's input. You need a way to translate what the user works with to the mini-language your program has developed. As we have discussed before, a parser is probably more work than you want to invest. My advice is that you introduce an environment variable for the screen. The screen will provide the information necessary for building the Reaction. You need to say what information this is. This is the point where you can customize your interface to match your assumptions. You can decide how many compounds on each side of the equation. My advice is that you can build compounds with a set number of molecules and each molecule is built from a drop down list with all of your elements and an edit box that accepts a natural number. I suggest that you do not have edit boxes for the coefficients, since those are what you are solving for. —SS] [Your GUI module will hide the details of working with a windowing system, widgets etc. The input module will use this GUI module. —SS]

6.4 Semantics

6.4.1 State Variables

Reaction : ReactionT
Mass : $\mathbb{R}$

6.4.2 Environment Variables

None. [You want an environment variable for your screen here. —SS]

6.4.3 Assumptions

None.

6.4.4 Access Routine Semantics

$input_1$(Chemical-Reaction):

- transition: Reaction := Chemical-Reaction
- output: None.
- exception: exc := (Chemical-Reaction $\notin$ ReactionT $\Rightarrow$ MismatchedInput)

[This is too simple for building the reaction. As discussed above, you need show how the user's input is translated into the terminology of your program. —SS]

$input_2$(Mass-Value):

- transition: Mass := Mass-Value
- output: None.
- exception: exc := (Mass-Value $\leq 0 \Rightarrow$ NegativeMassException)

[I think you need more than one mass don't you? —SS]

6.4.5 Local Functions

None.

7 MIS of Atomic Mass Module

7.1 Module

Atomic Mass

7.2 Uses

Reaction (Section 8)

7.3 Syntax

7.3.1 Exported Constants

None.

7.3.2 Exported Types

None.

7.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
Atomic-Mass	e: ElementT	AtomicMass: $\mathbb{R}$	e $\notin$ ElementT

7.4 Semantics

7.4.1 State Variables

None.

7.4.2 Environment Variables

Atomic Mass library file. [This is not how environment variables are used. I suggest you read Hoffman and Strooper (1995). For this module the entry should be "None." The only place you will have environment modules is when you are getting input from the screen (or a file), or outputting results to the screen (or a file). —SS]

7.4.3 Assumptions

None.

7.4.4 Access Routine Semantics

Atomic-Mass(e):

- transition: None.
- output: out := AtomicMass [AtomicMass on its own is a function. You need to call it with an argument in this case. Having said this, I don't think you really need a local function. You can define the output to match the semantics you have for the local function below. The indirection you have just makes the document more difficult to read. —SS]
- exception: exc := (e $\notin$ ElementT $\Rightarrow$ no-Atomic-Mass) [Since the input to the modules is of type ElementT, how is this condition ever going to be true for their to be an exception? —SS]

7.4.5 Local Functions

AtomicMass: ElementT $\rightarrow \mathbb{R}$
AtomicMass(e) $\equiv$ (e= H $\rightarrow$ 1.0079 | e= He $\rightarrow$ 4.002 |...)

8 MIS of ReactionT Module

8.1 Template Module

Reaction

8.2 Uses

Input (Section 6)

8.3 Syntax

8.3.1 Exported Constants

None.

8.3.2 Exported Types

ReactionT = ?

8.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
IsBalanced-Reaction	Reaction: ReactionT	B: Boolean	-
Balanced-Reaction	Reaction: ReactionT	Reaction*: ReactionT	($\neg$ IsBalanced-Reaction(Reaction*))

[You need a constructor for your ADT. The input to this ADT is of the same type as the ADT, but how do you create this input? —SS] [The isBalanced-Reaction should work with self to see if the reaction is itself is balanced. There is no need for an argument. —SS]

8.4 Semantics

8.4.1 State Variables

R : ReactionT

8.4.2 Environment Variables

None.

8.4.3 Assumptions

None.

8.4.4 Access Routine Semantics

IsBalanced-Reaction(Reaction):

- transition: None.
- output: out := ($\forall$(Number-of-Atoms-in-ChemicalEq(R[0],e) = Number-of-Atoms-in-ChemicalEq (R[1] ,e))$\Rightarrow$ True) [What is your dummy variable for the quantification? The $\Rightarrow$ is used for a conditional rule, but you do not have a conditional rule here. I think I know what you are trying to say, but mathematically this does not have the correct syntax. —SS]
- exception: None.

Balanced-Reaction(Reaction):

- transition: Reaction := Reaction*
- output: None.
- exception: exc := ($\neg$ IsBalanced-Reaction(Reaction*) $\Rightarrow$ not-balanced)

[This is where the most important part of your program happens. How do you balance a chemical reaction? You should at least have a statement that says what will potentially be changed in the reaction. You need to solve for the coefficients such that the reaction is balanced. You should also say what it takes for a unique solution. Ideally, you will solve so that the coefficients are all natural numbers. However, if you want to simplify, you can have the coefficients be real numbers. I'm not sure you have thought about how you are going to balance your equation. The following web-page gives a nice explanation of what is involved: `https://www.nayuki.io/page/chemical-equation-balancer-javascript` —SS]

8.4.5 Local Functions

1. Number-of-Atoms-in-Molecule: MoleculeT X ElementT $\rightarrow \mathbb{N}$
Number-of-Atoms-in-Molecule(m,e) $\equiv$ (m.element = e $\rightarrow$ m.number | m.element $\neq$ e $\rightarrow$ 0)

2. Number-of-Atoms-in-Compound: CompoundT X ElementT $\rightarrow \mathbb{N}$
Number-of-Atoms-in-Compound(C,e) $\equiv$ + (m : MoleculeT| m $\in$ C $\cdot$ Number-of-Atoms-in-Molecule(m,e))

3. Number-of-Atoms-in-Stoichiometric: StoichiometricT $\times$ ElementT $\rightarrow \mathbb{N}$
Number-of-Atoms-in-Stoichiometric(S,e) $\equiv$ (S.coefficient $\cdot$ Number-of-Atoms-in-Compound (S.Compound,e))

4. Number-of-Atoms-in-ChemicalEq: ChemicalEqT X ElementT $\rightarrow \mathbb{N}$
Number-of-Atoms-in-ChemicalEq (Ce,e) $\equiv$ + (S : StoichiometricT | S $\in$ Ce $\cdot$ Number-of-Atoms-in-Stoichiometric(S,e))

5. ElementsInCompoundT : CompoundT $\rightarrow$ set of ElementT.
ElementsInCompound(C) $\equiv \cup$ { m : MoleculeT | m $\in$ C $\cdot$ m.element}

6. ElementsInChemicalEq : ChemicalEqT $\rightarrow$ set of ElementT.
ElementsInChemicalEq(Ce) $\equiv \cup$ { C : CompoundT | C $\in$ Ce $\cdot$ ElementsInCompound(C)}

7. IsBalancedReactionForElement : ReactionT x ElementT $\rightarrow$ B
IsBalancedReactionForElement (R, e) $\equiv$ Number-of-Atoms-in-ChemicalEq(R[0],e) = Number-of-Atoms-in-ChemicalEq (R[1] ,e)

8. IsBalanced-Reaction : ReactionT $\rightarrow$ B
IsBalanced-Reaction(R) $\equiv \forall$ (e : ElementT | e $\in$ ElementsInChemicalEq(R[0]) $\cdot$ IsBalance-dReactionForElement(R, e))

9 MIS of Mass Calculation Module

9.1 Module

Mass

9.2 Uses

Input (Section 6) , Atomic Mass (Section 7) , Reaction (Section 8)

9.3 Syntax

9.3.1 Exported Constants

None.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Get-Molecular-weight	Atomic-Mass: $\mathbb{R}$ Number-of-Atoms-in-Compound: $\mathbb{N}$	Molecular-Weight:$\mathbb{R}$	-
Convert-Mass-to-Mole1	Mass1: $\mathbb{R}$ Molecular-Weight1: $\mathbb{R}$	Mole1: $\mathbb{R}$	Mass ≤ 0
Get-Mole-Ratio	Coefficient1 : $\mathbb{N}$ Coefficient2 : $\mathbb{N}$	Mole-Ratio: $\mathbb{N}$	-
Convert-MoleRatio-to-Mole2	Mole1: $\mathbb{R}$ Mole-Ratio: $\mathbb{R}$	Mole2: $\mathbb{R}$	-
Convert-Mole-to-Mass	Mole2: $\mathbb{R}$ Molecular-Weight2: $\mathbb{R}$	Mass2:$\mathbb{R}$	-

[Since you have a function for atomic mass of an element, you don't need atomic mass as an input for Get-Molecular-weight. Given a molecule, you can just calculate its weight. You have all of the information you need inside the module. —SS] [Several ADTs are missing. You need to define ElementT, MoleculeT, CompoundT, StoichiometricT and ChemicalEqT. When I mentioned ReactionT, that was just one example of the types that you need. —SS]

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

None

9.4.4 Access Routine Semantics

Get-Molecular-weight(Atomic-Mass,Number-of-Atoms-in-Compound):

- transition: None
- output: out $:=$ Atomic-Mass $\times$ Number-of-Atoms-in-Compound
- exception: None

Convert-Mass-to-Mole(Mass1, Molecular-Weight1):

- transition: None
- output: Mole1 $:= \frac{Mass1}{Molecular-Weight1}$
- exception: exc $:=$ (Mass1 $\leq 0 \Rightarrow$ NegativeMassException)

Get-Mole-Ratio(Coefficient2, Coefficient1):

- transition: None
- output: out $:= \frac{Coefficient2}{Coefficient1}$
- exception: None

Convert-MoleRatio-to-Mole(Mole1, Mole-Ratio):

- transition: None
- output: Mole2 $:= \frac{Mole1}{Mole-Ratio}$
- exception: None

Convert-Mole-to-Mass(Mole2, Molecular-Weight2):

- transition: None
- output: Mass2 $:= \frac{Mole2}{Molecular-weight2}$
- exception: None

9.4.5 Local Functions

None

10 MIS of GUI Module

10.1 Module

GUI [I discussed the purpose of the GUI module under the topic of the Input module. —SS]

10.2 Uses

Reaction (Section 8) , Mass (Section 9)

10.3 Syntax

10.3.1 Exported Constants

None.

10.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
GUI	-	Mass: $\mathbb{R}$ Reaction*: ReactionT	-

10.4 Semantics

Not Applicable.

[Further Notes on the missing parts of the design: —SS] [I also think you need an ADT for Sets. If you want to work without sets, you probably can, but they would nicely connect your SRS and your MIS. Defining a generic ADT for sets would be straightforward, but you don't even need to do that. The only sets you will have will be sets of elements, so you could define an ADT for sets of elements. You don't need to implement everything that sets have for operations, but you would need a test for equality. —SS] [If you find the idea of sets confusing and you don't want all of the ADTs (MoleculeT etc) I mentioned above, I believe that you can come up with a design that works. It is critical though that you have an ElementT. You can't just assume it. You also need a way to identify the coefficients that you are solving for. The goal for the next step is to be able to write code, but I don't think you know how you are going to balance your equations. Without coefficients you cannot explain what balancing an equation means. —SS]

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering.* Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

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.