Module Interface Specification for Lattice Boltzmann Solvers

Peter Michalski

November 24, 2019

1 Revision History

Date	Version	Notes
Nov. 25, 2019	1.0	Initial Document

2 Symbols, Abbreviations and Acronyms

See CA Documentation for Lattice Boltzmann Solvers (Michalski, a).

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **2**

6 MIS of M2: System Control Module **3**
- 6.1 Module . . . 3
- 6.2 Uses . . . 3
- 6.3 Syntax . . . 3
  - 6.3.1 Exported Constants . . . 3
  - 6.3.2 Exported Access Programs . . . 3
- 6.4 Semantics . . . 3
  - 6.4.1 State Variables . . . 3
  - 6.4.2 Environment Variables . . . 3
  - 6.4.3 Assumptions . . . 4
  - 6.4.4 Access Routine Semantics . . . 4
  - 6.4.5 Local Functions . . . 4
  - 6.4.6 Local Function Semantics . . . 4

7 MIS of M3: Input Reading Module **5**
- 7.1 Module . . . 5
- 7.2 Uses . . . 5
- 7.3 Syntax . . . 5
  - 7.3.1 Exported Constants . . . 5
  - 7.3.2 Exported Access Programs . . . 5
- 7.4 Semantics . . . 5
  - 7.4.1 State Variables . . . 5
  - 7.4.2 Environment Variables . . . 5
  - 7.4.3 Assumptions . . . 5
  - 7.4.4 Access Routine Semantics . . . 6
  - 7.4.5 Local Functions . . . 6

8 MIS of M4: Input Checking Module **7**
- 8.1 Module . . . 7
- 8.2 Uses . . . 7
- 8.3 Syntax . . . 7
  - 8.3.1 Exported Constants . . . 7

8.3.2 Exported Access Programs . . . 7
8.4 Semantics . . . 7
8.4.1 State Variables . . . 7
8.4.2 Environment Variables . . . 8
8.4.3 Assumptions . . . 8
8.4.4 Access Routine Semantics . . . 8
8.4.5 Local Functions . . . 8

9 MIS of M5: LBM Control Module 9
9.1 Module . . . 9
9.2 Uses . . . 9
9.3 Syntax . . . 9
9.3.1 Exported Constants . . . 9
9.3.2 Exported Access Programs . . . 9
9.4 Semantics . . . 9
9.4.1 State Variables . . . 9
9.4.2 Environment Variables . . . 9
9.4.3 Assumptions . . . 9
9.4.4 Access Routine Semantics . . . 9
9.4.5 Local Functions . . . 10

10 MIS of M6: Streaming Module 11
10.1 Module . . . 11
10.2 Uses . . . 11
10.3 Syntax . . . 11
10.3.1 Exported Constants . . . 11
10.3.2 Exported Access Programs . . . 11
10.4 Semantics . . . 11
10.4.1 State Variables . . . 11
10.4.2 Environment Variables . . . 11
10.4.3 Assumptions . . . 11
10.4.4 Access Routine Semantics . . . 11
10.4.5 Local Functions . . . 12
10.4.6 Local Function Semantics . . . 12

11 MIS of M7: Collision Module 13
11.1 Module . . . 13
11.2 Uses . . . 13
11.3 Syntax . . . 13
11.3.1 Exported Constants . . . 13
11.3.2 Exported Access Programs . . . 13
11.4 Semantics . . . 13
11.4.1 State Variables . . . 13

11.4.2 Environment Variables 13
11.4.3 Assumptions 13
11.4.4 Access Routine Semantics 14
11.4.5 Local Functions 14
11.4.6 Local Function Semantics 14

12 MIS of M8: Problem Module **15**
12.1 Module 15
12.2 Uses 15
12.3 Syntax 15
12.3.1 Exported Constants 15
12.3.2 Exported Access Programs 15
12.4 Semantics 15
12.4.1 State Variables 15
12.4.2 Environment Variables 15
12.4.3 Assumptions 15
12.4.4 Access Routine Semantics 16
12.4.5 Local Functions 16

13 MIS of M9: Lattice Module **17**
13.1 Module 17
13.2 Uses 17
13.3 Syntax 17
13.3.1 Exported Constants 17
13.3.2 Exported Access Programs 17
13.4 Semantics 17
13.4.1 State Variables 17
13.4.2 Environment Variables 17
13.4.3 Assumptions 17
13.4.4 Access Routine Semantics 17
13.4.5 Local Functions 18

14 MIS of M10: Boundary Module **19**
14.1 Module 19
14.2 Uses 19
14.3 Syntax 19
14.3.1 Exported Constants 19
14.3.2 Exported Access Programs 19
14.4 Semantics 19
14.4.1 State Variables 19
14.4.2 Environment Variables 19
14.4.3 Assumptions 19
14.4.4 Access Routine Semantics 19

14.4.5 Local Functions 20

15 MIS of M11: Image Rendering Module **21**
15.1 Module 21
15.2 Uses 21
15.3 Syntax 21
15.3.1 Exported Constants 21
15.3.2 Exported Access Programs 21
15.4 Semantics 21
15.4.1 State Variables 21
15.4.2 Environment Variables 21
15.4.3 Assumptions 21
15.4.4 Access Routine Semantics 21
15.4.5 Local Functions 22

16 MIS of M12: Data Structure Module **23**
16.1 Module 23
16.2 Uses 23
16.3 Syntax 23
16.3.1 Exported Constants 23
16.3.2 Exported Access Programs 23
16.4 Semantics 23
16.4.1 State Variables 23
16.4.2 State Invariant 23
16.4.3 Assumptions 23
16.4.4 Access Routine Semantics 23
16.4.5 Local Functions 24

17 MIS of M13: Input Types Module **25**
17.1 Module 25
17.2 Uses 25
17.3 Syntax 25
17.3.1 Exported Constants 25
17.3.2 Exported Access Programs 25
17.4 Semantics 25
17.4.1 State Variables 25
17.4.2 State Invariant 25
17.4.3 Assumptions 25
17.4.4 Access Routine Semantics 25
17.4.5 Local Functions 25

18 Appendix **27**

3 Introduction

The following document details the Module Interface Specifications for Lattice Boltzmann Solvers, which provides a library of services based on Lattice Boltzmann Methods (LBM). LBM are a family of fluid dynamics algorithms for simulating single-phase and multiphase fluid flows, often incorporating additional physical complexities (Chen and Doolen, 1998).

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found here (Michalski, b).

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1999), 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 (1999). 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 Lattice Boltzmann Solvers.

Data Type	**Notation**	**Description**
string	string	single or multiple symbols or digits
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

The specification of Lattice Boltzmann Solvers 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, Lattice Boltzmann Solvers 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.

5 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	M2: System Control Module M3: Input Reading Module M4: Input Checking Module M5: LBM Control Module M6: Streaming Module M7: Collision Module M8: Problem Module M9: Lattice Module M10: Boundary Module
Software Decision Module	M11: Image Rendering Module M12: Data Structure Module M13: Input Types Module

Table 1: Module Hierarchy

6 MIS of M2: System Control Module

The secret of this module is the algorithm to control Lattice Boltzmann Solvers.

6.1 Module

SystemControl

6.2 Uses

- Hardware Hiding
- Input Reading (Section 7)
- LBM Control (Section 9)
- Problem Parameter (Section 12)
- Image Rendering (Section 15)
- Data Structure (Section 16)

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

N/A

6.4 Semantics

6.4.1 State Variables

inputData: DataStructure
problemData: $\mathbb{R}$ & $\mathbb{N}$
imageData: $\mathbb{R}$
imageOut: PNG

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

The user has run the Lattice Boltzmann Solvers program.

6.4.4 Access Routine Semantics

N/A

6.4.5 Local Functions

Name	**In**	**Out**	**Exceptions**
mainFunction	-	-	-

6.4.6 Local Function Semantics

mainFunction():

- transition: out := N/A
- exception: N/A

The function calls other modules of Lattice Boltzmann Solvers to solve the given problem. The function calls InputReading.inputArray(), followed by ProblemParameter.formatInput() and LBMControl.performLBM(). Finally the module calls ImageRendering.imageFunc() and sends the output of that function to the hardware.

7 MIS of M3: Input Reading Module

The secret of this module is the algorithm that gathers the input data.

7.1 Module

InputReading

7.2 Uses

- Hardware Hiding
- Input Checking (Section 8)
- Data Structure (Section 16)

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
inputArray	-	inputData: DataStructure	NotFound, ErrorRead

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

InputLocation (string): "./Input/input.txt"

7.4.3 Assumptions

The System Control Module M2 (Section 6) has called InputReading.inputArray().

7.4.4 Access Routine Semantics

inputArray():

- transition: $out := +(\text{inputData}[\text{key}][\text{value}] \Leftarrow ./\text{Input}/\text{input.txt})$
- exception: $\text{input.txt} \notin ./\text{Input}/ \Rightarrow \text{NotFound}$
- exception: $(\text{inputData} \Leftarrow ?) \Rightarrow \text{ErrorRead}$

The function will read all lines from the input file and place each value into inputData.

7.4.5 Local Functions

N/A

8 MIS of M4: Input Checking Module

This secret of this module is the algorithm that checks if input values fall within allowable parameters.

8.1 Module

InputChecking

8.2 Uses

- Data Structure (Section 16)

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
verifyInputs	inputData: DataStructure	-	OutBounds, UnknwnParm

8.4 Semantics

8.4.1 State Variables

The set of acceptableRanges is:
LIBRARY: set of $\mathbb{N}$: $\{1\}$ - Libraries are associated by numbers in the program. The numbers are available for reference in the User Guide.
PROBLEM: set of $\mathbb{N}$: $\{1\}$ - Problems are associated by numbers in the program. The numbers are available for reference in the User Guide.
DIMENSIONS: set of $\mathbb{N}$: $\{2\}$
VEL_DIRS: set of $\mathbb{N}$: $\{9\}$
REYNOLDS_MIN: $\mathbb{R}$: $\{0.001\}$
REYNOLDS_MAX: $\mathbb{R}$: $\{5000\}$
DENSITY_MIN: $\mathbb{R}$: $\{0.0708\}$
DENSITY_MIN: $\mathbb{R}$: $\{13.6\}$
BULK_VIS_MIN: $\mathbb{R}$: $\{0.0001\}$
BULK_VIS_MIN: $\mathbb{R}$: $\{20000\}$
SHEAR_VIS_MIN: $\mathbb{R}$: $\{0.001\}$
SHEAR_VIS_MIN: $\mathbb{R}$: $\{20000\}$

$TIME\_MIN$: $\mathbb{N}$: $\{1\}$

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

The Input Reading Module M3 (Section 7) has called InputChecking.verifyInputs().

8.4.4 Access Routine Semantics

verifyInputs(inputData):

- output: N/A
- exception: (inputData[value] $>$ acceptableRanges) $\cup$ (inputData[value] $<$ acceptableRanges) $\Rightarrow$ OutBounds
- exception: inputData[key] $\notin$ inputTypes $\Rightarrow$ UnknwnParm

The function will iterate through each inputValue InputTypes key and check if the key is known to the program, as well as check if associated values of known keys fall within an acceptable range of the state variables.

8.4.5 Local Functions

N/A

9 MIS of M5: LBM Control Module

The secret of this module is the algorithm which controls the LBM library.

9.1 Module

LBMControl

9.2 Uses

- Streaming (Section 10)
- Collision (Section 11)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
performLBM	inputData: DataStructure, problemData: $\mathbb{R}$ & $\mathbb{N}$	imageData: $\mathbb{R}$	-

9.4 Semantics

9.4.1 State Variables

imageData ($\mathbb{R}$)

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

The System Control Module M2 (Section 6) has called LBMControl.performLBM().

9.4.4 Access Routine Semantics

performLBM(inputData, problemData):

- transition: $out := \text{imageData} = \mathbb{R} \Leftarrow \text{Streaming.streamingFunc}(), \mathbb{R} \Leftarrow \text{Collision.collisionFunc}()$
- exception: N/A

The function will calculate the vorticity vector values, iterating through each velocity direction, calling the streaming and collision module functions.

9.4.5 Local Functions

N/A

10 MIS of M6: Streaming Module

The secret of this module is the algorithm to calculate the streaming pf particles.

10.1 Module

Streaming

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
streamingFunc	Velocityi, Time, PositionVector, Force, Mass: $\mathbb{R}$	$\mathbb{R}$	NAN

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

maxVariableSize = The maximum allowable value held in the variable.
minVariableSize = The minimum allowable value held in the variable.

10.4.3 Assumptions

The LBM Control Module M5 (Section 9) has called streaming.streamingFunc().

10.4.4 Access Routine Semantics

streamingFunc(Velocityi, Time, PositionVector, Force, Mass):

- transition: $out := f_i(\mathbf{x} + \mathrm{e}_i dt, t + dt) - f_i(\mathbf{x}, t)$

- exception: $(f_i(\mathbf{x}+\mathrm{e}_i dt, t+dt) - f_i(\mathbf{x}, t) > \text{maxVariableSize}) \cup (f_i(\mathbf{x}+\mathrm{e}_i dt, t+dt) - f_i(\mathbf{x}, t) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the streaming step value for each velocity direction (i).

10.4.5 Local Functions

Name	**In**	**Out**	**Exceptions**
pdfFunc ($f(\mathbf{x}, \mathrm{e}, t)$)	Velocityi, Time, PositionVector, Force, Mass: $\mathbb{R}$	$\mathbb{R}$	NAN

10.4.6 Local Function Semantics

pdfFunc(Velocityi, Time, PositionVector, Force, Mass):

- transition: $out := f(\mathbf{x}, \mathrm{e}, t) = f(\mathbf{x} + \mathrm{e}dt, \mathrm{e} + \frac{F}{kg}dt, t + dt)$

- exception: $(f(\mathbf{x}+\mathrm{e}dt, \mathrm{e}+\frac{F}{kg}dt, t+dt)) > \text{maxVariableSize}) \cup (f(\mathbf{x}+\mathrm{e}dt, \mathrm{e}+\frac{F}{kg}dt, t+dt)) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function finds the probability that a particle is at position $\mathbf{x}$ and has velocity e at time t.

11 MIS of M7: Collision Module

The secret of this module is the algorithm to calculate the collision of particles.

11.1 Module

Collision

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
collisionFunc	RelaxationRate, Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

maxVariableSize = The maximum allowable value held in the variable.
minVariableSize = The minimum allowable value held in the variable.

11.4.3 Assumptions

The LBM Control Module M5 (Section 9) has called Collision.streamingFunc().

11.4.4 Access Routine Semantics

collisionFunc(RelaxationRate, Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := \frac{1}{\tau}(f_i^{eq} - f_i)$
- exception: $(\frac{1}{\tau}(f_i^{eq} - f_i) > \text{maxVariableSize}) \cup (\frac{1}{\tau}(f_i^{eq} - f_i) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the collision step value for each velocity direction (i).

11.4.5 Local Functions

Name	In	Out	Exceptions
edfFunc (f_i^{eq})	Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN
relFunc (f_i)	Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN

11.4.6 Local Function Semantics

edfFunc(Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := f_i^{eq} = w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4})$
- exception: $(w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4}) > \text{maxVariableSize}) \cup (w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4}) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the equilibrium distribution for each velocity direction (i).

relFunc(Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := f_i = f_i - \frac{1}{\tau}(f_i^{eq} - f_i)$
- exception: $(f_i - \frac{1}{\tau}(f_i^{eq} - f_i) > \text{maxVariableSize}) \cup (f_i - \frac{1}{\tau}(f_i^{eq} - f_i) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the relaxation update for each velocity direction (i).

12 MIS of M8: Problem Module

The secret of this module is the structure of the LBM input parameters.

12.1 Module

ProblemParameter

12.2 Uses

- Lattice (Section 13)
- Boundary (Section 14)
- Data Structure (Section 16)

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
formatInput	inputData: DataStructure	problemData: $\mathbb{R}$ & $\mathbb{N}$	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

The System Control Module M2 (Section 6) has called ProblemParameter.formatInput().

12.4.4 Access Routine Semantics

formatInput(inputData):

- transition: *out* := problemData = $\mathbb{N} \in$ inputData, $\mathbb{R} \Leftarrow$ Boundary.getBoundary(), $\mathbb{R} \Leftarrow$ Lattice.getWeights()
- exception: N/A

The function will set up the structure for the LBM input parameters based on the library that the user has requested. The function will use the Lattice and Boundary module functions.

12.4.5 Local Functions

N/A

13 MIS of M9: Lattice Module

The secret of this module is the structure of the lattice model.

13.1 Module

Lattice

13.2 Uses

- Data Structure (Section 16)

13.3 Syntax

13.3.1 Exported Constants

parameterValues := (9, 4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36)
The above holds the coefficient weights for velocity directions of a Q9 model.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
getWeights	inputData: DataStructure	$\mathbb{R}$	NoLattice

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

The Problem Module M8 (Section 12) has called Lattice.getWeights().

13.4.4 Access Routine Semantics

getWeights(inputData):

- transition: out := Weighti ($\mathbb{R}$) $\in$ parameterValues
- exception: inputData[VelocityDirections] $\notin$ parameterValues $\Rightarrow$ NoLattice

The function sets coefficient weight data for the input lattice model, and returns an error if the weight data is not available for the input lattice model.

13.4.5 Local Functions

N/A

14 MIS of M10: Boundary Module

The secret of this module is the structure of the model boundary.

14.1 Module

Boundary

14.2 Uses

- Data Structure (Section 16)

14.3 Syntax

14.3.1 Exported Constants

N/A

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
getBoundary	inputData: DataStructure	$\mathbb{R}$	-

14.4 Semantics

14.4.1 State Variables

N/A

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

The Problem Module M8 (Section 12) has called Boundary.getBoundary().

14.4.4 Access Routine Semantics

getBoundary(inputData):

- transition: $out := \mathbb{R} \in$ DataStructure
- exception: N/A

The function returns the boundary from the DataStructure library.

14.4.5 Local Functions

N/A

15 MIS of M11: Image Rendering Module

The algorithm to convert the LBM output into an image.

15.1 Module

ImageRendering

15.2 Uses

N/A

15.3 Syntax

15.3.1 Exported Constants

N/A

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
imageFunc	imageData: $\mathbb{R}$	imageOut: PNG	IncorrectFormat

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

The System Control Module M2 (Section 12) has called ImageRendering.imageFunc() of this module.

15.4.4 Access Routine Semantics

imageFunc(imageData):

- transition: *out* := imageOut: PNG
- exception: (imageData[?] | ? $\neq \mathbb{R} \Rightarrow$ IncorrectFormat) (? is not a real number)

The function converts the information from the LBM algorithm output (for example the vorticity vector values) into an image format using an external image library.

15.4.5 Local Functions

N/A

16 MIS of M12: Data Structure Module

The format of a non-primitive data structure for input variables.

16.1 Module

DataStructure

16.2 Uses

- Input Types (Section 17)

16.3 Syntax

16.3.1 Exported Constants

N/A

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	-	-	-
add	InputTypes, $\mathbb{R}$	-	IncorrectFormat
getElm	InputTypes	$\mathbb{R}$	DoesNotExist

16.4 Semantics

16.4.1 State Variables

s: set of tuple of (key: InputType, value: $\mathbb{R}$)

16.4.2 State Invariant

N/A

16.4.3 Assumptions

DataStructure.init() is called before any other access program.

16.4.4 Access Routine Semantics

init():

- transition: $s := \{\}$
- exception: N/A

add(InputTypes, $\mathbb{R}$):

- transition: $s := s \cup \{\langle$ InputTypes, $\mathbb{R}$ $\rangle\}$
- exception: ($\langle$ InputTypes, ?$\rangle \in s \Rightarrow$ IncorrectFormat) (? is not a real number)

remove(InputTypes):

- transition: $s := s$ - $\{\langle$ InputTypes, $\mathbb{R}$ $\rangle\}$ where $\langle$ InputTypes, $\mathbb{R}$ $\rangle \in s$
- exception: ($\langle$ InputTypes, $\mathbb{R}$ $\rangle \notin s \Rightarrow$ DoesNotExist)

getElm(InputTypes):

- transition: $out := \langle$ InputTypes, $\mathbb{R}$ $\rangle \in s$
- exception: ($\langle$ InputTypes, $\mathbb{R}$ $\rangle \notin s \Rightarrow$ DoesNotExist)

16.4.5 Local Functions

N/A

17 MIS of M13: Input Types Module

The input types for the non-primitive data structures.

17.1 Module

InputTypes

17.2 Uses

N/A

17.3 Syntax

17.3.1 Exported Constants

InputTypes (string) = {Library, Problem, Dimensions, VelocityDirections, ReynoldsNumber, Density, BulkViscosity, ShearViscosity, Time, Density, Viscosity, AccelerationRate, SpeedSound, Velocity, Force, Mass, CrossSectionalArea, MacroscopicVelocity, RelaxationRate}

17.3.2 Exported Access Programs

N/A

17.4 Semantics

17.4.1 State Variables

N/A

17.4.2 State Invariant

N/A

17.4.3 Assumptions

The Data Structure Module M12 (Section 16) will only use a set of the above InputTypes.

17.4.4 Access Routine Semantics

N/A

17.4.5 Local Functions

N/A

References

Shiyi Chen and Gary D Doolen. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

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

Daniel Hoffman and Paul Strooper. Software Design, Automated Testing, and Maintenance–A Practical Approach. 1999.

Peter Michalski. Lattice Boltzmann Solvers - CA, a. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/SRS/CA.pdf.

Peter Michalski. Lattice Boltzmann Solvers, b. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers.

18 Appendix

Table 2: Possible Exceptions

Message ID	Error Message
DoesNotExist	Error: The input type does not exist.
ErrorRead	Error: Could not read the input file.
IncorrectFormat	Error: The format is not a real number.
NAN	Error: The calculated result is not a number.
NoLattice	Error: The chosen velocity directions do not have a known lattice structure.
NotFound	Error: Input file not found.
OutBounds	Error: The input file parameter X is out of bounds. Please see the User Guide.
UnknwnParm	Error: The parameter X is not known to the system. Please see the User Guide.