

CAS 741, CES 741 (Development of Scientific Computing Software)

Fall 2019

17 Math Review Plus MIS Example

Dr. Spencer Smith

Faculty of Engineering, McMaster University

October 31, 2019



Math Review

- Administrative details
- Questions?
- SRS feedback
- Motivating example: Chemical reactions
- Math review introduction
- Review of sets, relations and functions
- Review of logic
- Review of types, sets, sequence and tuples
- Multiple assignment statement
- Conditional rules
- Finite State Machines
- Example MIS: 2D Data

Administrative Details

- SRS grades on Avenue
- If follow feedback, final doc grades will be higher
- GitHub issues for colleagues
 - ▶ Assigned 1 colleague (see Repos.xlsx in repo)
 - ▶ Provide at least 3 issues on their MG
 - ▶ Grading as before
 - ▶ Due by Thursday, Nov 8, 11:59 pm
- MG marking scheme in Avenue
- MIS marking scheme in Avenue
- Updated MIS template in CAS 741 repo (soon)

Administrative Details: Report Deadlines

MG + MIS	Week 10	Nov 25
Final Documentation	Week 14	Dec 9

- The written deliverables will be graded based on the repo contents as of 11:59 pm of the due date
- If you need an extension, please ask
- Two days after each major deliverable, your GitHub issues will be due
- Domain expert code due 1 week after MIS deadline

Administrative Details: Presentations

MG + MIS Syntax Present	Week 9	Week of Nov 4
MIS Semantics Present	Week 11	Week of Nov 18
Unit VnV or Impl. Present	Week 12/13	Week of Nov 28

- Informal presentations with the goal of improving everyone's written deliverables
- Domain experts and secondary reviewers (and others) will ask questions (listed in Repos.xlsx file)

Administrative Details: Presentation Schedule

- MIS Syntax + Semantics Present
 - ▶ Monday: Zhi, Peter, *Sasha*
 - ▶ Thursday: Sharon, Ao, ?
- Unit VnV Plan or Impl. Present
 - ▶ Monday: Bo, Sasha, ?
 - ▶ Thursday: Zhi, Peter, Ao, ?

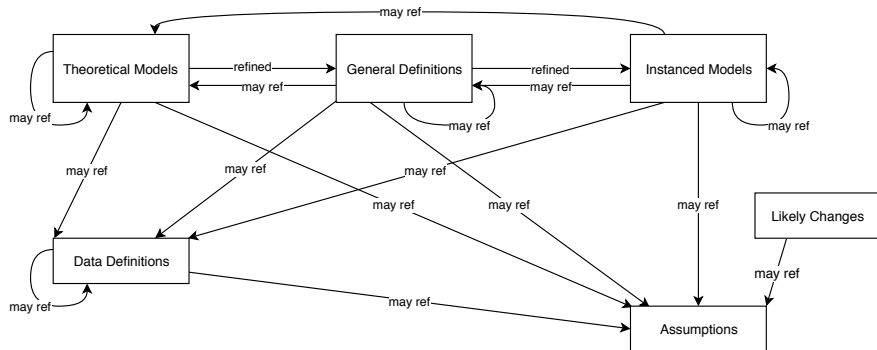
Optional presentation in italics.

Room for more volunteers. :-)

Questions?

- Questions about Module Guides?
- Questions about MIS?

SRS Feedback: Relationships Between Parts



SRS Feedback

- *.DS_Store should be in .gitignore
- L^AT_EX and formatting rules
 - ▶ Variables are italic, everything else not, includes subscripts (link to document)
 - ▶ Conventions
 - ▶ Watch out for implied multiplication
 - ▶ Use BibTeX
 - ▶ Use cross-referencing
- Grammar and writing rules
 - ▶ Acronyms expanded on first usage (not just in table of acronyms)
 - ▶ “In order to” should be “to”

SRS Feedback on Applying Template

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be “not applicable” for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- “Referenced by” implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract
- For families the notion of binding time should be introduced
- Think of families as a library, not as a single program

Mathematical Review: Introduction

- The material in these slides should hopefully be review, or reasonably easy to pick up
- Shows the simple mathematics that can be used to build your MIS
- Shows a syntax that you can use
- The presentation follows [2] (Chapter 3) and [1]

Sets, Relations and Functions

- A set is an unordered collection of elements
- A binary relation is a set of ordered pairs
- A function is a relation in which each element in the domain appears exactly once as the first component in the ordered pair

Sets

- An element either belongs to a set or it does not
- $x \in S$ versus $x \notin S$
- Defining a set
 - ▶ Enumerate $\{x_1, x_2, x_3, \dots, x_n\}$
 - ▶ Logical condition (rule) $\{x | p(x)\}$
 - ▶ Notation from [1] $\{x : X | R : E\}$
 - ▶ An integer range $[2..4] = \{2, 3, 4\}$, $[7..4] = \{\}$
- Examples
 - ▶ $S = \{1, 7, 6\}$
 - ▶ $S = \{x | x \text{ is an integer between 1 and 4 inclusive} \}$
 - ▶ $S = \{x : \mathbb{N} | 0 \leq x \leq 4 : x \}$
- Does $\{1, 7, 6\} = \{7, 1, 6\}$?

Relations

- Let $\langle x, y \rangle$ denote an ordered pair
 - ▶ $\text{dom}(R) = \{x \mid \langle x, y \rangle \in R\}$
 - ▶ $\text{ran}(R) = \{y \mid \langle x, y \rangle \in R\}$
- Defining a relation
 - ▶ Enumerate $\{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 2, 3 \rangle\}$
 - ▶ Rule $\{\langle x, y \rangle \mid x \text{ and } y \text{ are integers and } x < y\}$
 - ▶ $\{x, y : \mathbb{N} \mid x < y : \langle x, y \rangle\}$

Functions

- Let $\langle x, y \rangle$ denote an ordered pair
- Each element of the domain is associated with a unique element of the range
- Defining a function
 - ▶ Enumerate $\{\langle 0, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 3 \rangle\}$
 - ▶ Rule $\{\langle x, y \rangle \mid x \text{ and } y \text{ are integers and } y = x^2\}$
- Notation
 - ▶ $f(a) = b$ means $\langle a, b \rangle \in f$
 - ▶ $f(x) = x^2$
 - ▶ $f : T_1 \rightarrow T_2$
 - ▶ $\{\langle \langle x_1, x_2 \rangle, y \rangle \mid x_1, x_2 \text{ are integers and } y = x_1 + x_2\}$
- Is $\{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 2, 3 \rangle\}$ a function?
- Is $\{\langle x, y \rangle \mid x \text{ and } y \text{ are integers and } y^2 = x\}$?

Logic

- A logical expression is a statement whose truth values can be determined ($6 < 7?$)
- Truth values are either *true* or *false*
- Complex expressions are formed from simpler ones using logical connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$)
- Truth tables
- Evaluation
 - ▶ Decreasing order of precedence: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
 - ▶ Evaluate from left to right
 - ▶ Use rules of boolean algebra

Quantifiers

- Variables are often used inside logical expressions
- Variables have types
- A type is a set of values from which the variable can take its value
- Often quantify a logical expression over a given variable
 - ▶ Universal quantification
 - ▶ Existential quantification

Quantifiers Continued

- Prefer [1, p. 143] notation for quantification (and set building)
- $(*x : X | R : P)$ means application of the operator $*$ to the values P for all x of type X for which range R is true. In the above equations, the $*$ operators include \forall , \exists and $+$ are used
- Example on next slide for rank function specification

$$\text{rank}(a, A) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$$

$$\text{rank}(a, A) \equiv \text{avg}(\text{indexSet}(a, \text{sort}(A)))$$

$$\text{indexSet}(a, B) : \mathbb{R} \times \mathbb{R}^n \rightarrow \text{set of } \mathbb{N}$$

$$\text{indexSet}(a, B) \equiv \{j : \mathbb{N} | j \in [1..|B|] \wedge B_j = a : j\}$$

$$\text{sort}(A) : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

$$\text{sort}(A) \equiv B : \mathbb{R}^n, \text{ such that}$$

$$\forall(a : \mathbb{R} | a \in A : \exists(b : \mathbb{R} | b \in B : b = a) \wedge \text{count}(a, A) = \text{count}(b, B)) \wedge \forall(i : \mathbb{N} | i \in [1..|A| - 1] : B_i \leq B_{i+1})$$

$$\text{count}(a, A) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{N}$$

$$\text{count}(a, A) \equiv + (x : \mathbb{R} | x \in A \wedge x = a : 1)$$

$$\text{avg}(C) : \text{set of } \mathbb{N} \rightarrow \mathbb{R}$$

$$\text{avg}(C) \equiv + (x : \mathbb{N} | x \in C : x) / |C|$$

Quantifiers Continued

- Bound variables appear in the scope of the quantifier
- Free variables are not bound to any quantifier
- Free variables in an expression often mean that we cannot determine the truth value of the expression

Types, Sets, Sequence and Tuples

- A type is a set of values, so any precisely defined set is a type
- Primitive types are often integer, boolean, character, string and real
- Types can include functions ($T_1 \rightarrow T_2$)
- User-defined types
 - ▶ The set of values has to be given
 - ▶ Often use type constructors
- Useful type constructors
 - ▶ Set
 - ▶ Sequence
 - ▶ Tuple

Types

- Specify the type of a variable
 - ▶ $x_1, x_2, \dots, x_n : T$
 - ▶ $x : \textit{integer}$
 - ▶ $a, b, c : \textit{string}$
- Type definition
 - ▶ $T = d$
 - ▶ $\textit{float} = \textit{real}$
 - ▶ $\textit{colour} = \{\textit{red}, \textit{white}, \textit{blue}\}$
 - ▶ $\textit{testtype} = \{\textit{uniaxial}, \textit{biaxial}, \textit{shear}\}$
 - ▶ $x : \textit{testtype}$
 - ▶ $\textit{motion}T = \{\textit{forward}, \textit{backward}, \textit{stop}\}$

Primitive Types

- Integer

- ▶ $\{\dots - 2, -1, 0, 1, 2, \dots\}$
- ▶ $+, -, \times, /$
- ▶ $=, \neq$
- ▶ $<, \leq, \geq, >$

- Real

- ▶ $\{\textit{all real numbers}\}$
- ▶ $+, -, \times, /, \sin(), \cos(), \exp()$ etc.
- ▶ $=, \neq$
- ▶ $<, \leq, \geq, >$

Primitive Types Continued

- Boolean type
 - ▶ $\{true, false\}$
 - ▶ $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Char type
 - ▶ Set of ASCII characters
 - ▶ Character values appear in quotes $'a', 'b', 'c'$, etc.
 - ▶ $=, \neq$

Primitive Types Continued

- String type
 - ▶ All finite sequences of characters
 - ▶ String constants are in double quotes `"abc"`
 - ▶ $s[i..j]$ is the substring of s from position i to position j
 - ▶ $s_1 || s_2$ concatenates strings s_1 and s_2
 - ▶ $=, \neq$ for is equal and not equal
 - ▶ \in, \notin for is member and not a member
 - ▶ $s[i]$ is the i th character of s
 - ▶ $|s|$ is the length of s
 - ▶ Positions in strings are zero relative

Sets

- A set is an unordered collection of elements of the same type
- Declare a set of type T as *set of T*
- Example
 - ▶ $T = \text{set of } \{\text{red}, \text{green}, \text{blue}\}$ defines type T as the power set of $\{\text{red}, \text{green}, \text{blue}\}$
 - ▶ $x : \text{set of integer}$
- What are some possible values for $x : \text{set of integer}$?

Operations on Sets

- \cup union
- \cap intersection
- $-$ difference
- \times Cartesian product
- $=, \neq$ equal, not equal
- \in, \notin member, non-member
- $|s|$ size of set s

Sequences

- A sequence is an ordered collection of elements of the same type
 - ▶ Elements can occur more than once
 - ▶ Sometimes referred to as a list
 - ▶ Similar to an array
- Declare a sequence of type T by *sequence of T*
- $\langle x_0, x_1, \dots, x_n \rangle$ for $n \geq 0$ for a sequence with elements x_0, x_1, \dots, x_n
- $\langle \rangle$ is the empty sequence
- Position in a sequence is zero relative

Sequences Continued

- Examples
 - ▶ $T = \text{sequence of } \{\text{red}, \text{green}, \text{blue}\}$ defines the type T as the set of all sequences of elements from $\{\text{red}, \text{green}, \text{blue}\}$
 - ▶ $x : \text{sequence of integer}$
- Fixed-length sequence of type T with length l
 - ▶ $\text{sequence } [l] \text{ of } T$
 - ▶ l is a positive integer
 - ▶ $\text{sequence } [l_1, l_2, \dots, l_n] \text{ of } T$ is a shorthand for $\text{sequence } [l_1] \text{ of sequence } [l_2] \text{ of } \dots \text{sequence } [l_n] \text{ of } T$

Operations on Sequences

- $s[i..j]$ is the subsequence of s from position i to position j
- $s[i..j]$ is undefined if $i \notin [0..|s| - 1] \vee j \notin [0..|s| - 1]$
- $s_1 || s_2$ concatenates sequences s_1 and s_2
- $=, \neq$ for is equal and not equal
- \in, \notin for is member and not a member
- $s[i]$ is the i th element of s
- $s[i]$ is undefined if $i \notin [0..|s| - 1]$
- $|s|$ is the length of s
- A string is a sequence of characters

Tuples

- A tuple is a collection of elements of possibly different types
- Each tuple has one or more fields
- Each field has a unique identifier called the field name
- Similar to a record or a structure
- To declare a tuple use
 - ▶ *tuple of* $(f_1 : T_1, f_2 : T_2, \dots, f_n : T_n)$ with $n \geq 1$
 - ▶ f_i is the name of the i th field
 - ▶ T_i is the type of the i th field
 - ▶ *tuple of* $(f_1, f_2, \dots, f_n : T)$ if all fields are of the same type

Example Tuples

- Examples
 - ▶ *pair* = tuple of (*id* : integer, *val* : string)
 - ▶ *experimentT* = tuple of (*b_{cond}* : *bcondT*, *control* : *controlT*)
- Define the value of a tuple by using an expression of the form
 - ▶ $\langle x_1, x_2, \dots, x_n \rangle$
 - ▶ $\langle 4, \text{"cat"} \rangle$ is a value of type *pair*

Operations on Tuples

- $=, \neq$ equal, not equal
- $t.f$ is the value of field f of tuple t

Using Type Constructors

- $bcondT = \{uniaxial, biaxial, multiaxial, shear\}$
- $controlT = \{load_controlled, displacement_controlled\}$
- $experimentT = \text{tuple of } (b_{cond} : bcondT, control : controlT)$
- $experiment : experimentT$
- $directionT = \{clockwise, counterclockwise\}$
- $powerT = [MIN_POWER \dots MAX_POWER]$
- $motorT = \text{tuple of } (powerOn : Boolean, direction : directionT, powerLevel : powerT)$

Multiple Assignment Statement

- $v_1, v_2, \dots, v_n := e_1, e_2, \dots, e_n$ with $n \geq 1$
- The v_i s are distinct variables and each e_i is an expression of the same type as v_i
- Compute the values of all the expression e_i and then assign these values simultaneously
- Example
 - ▶ $x, y := 0, 10$
 - ▶ $x, y := 10, x$
 - ▶ $x, y := y, x$
- Convenient for defining the meaning of pieces of code
- Use as a function on the state space of a program

Conditional Rules

- $(c_1 \Rightarrow r_1 | \dots | c_n \Rightarrow r_n)$, where $n \geq 1$
- c_i s are the logical expressions
- r_i s are the rules
- $c_i \Rightarrow r_i$ is the i th component of the rule
- The first c_i that evaluates to true applies rule r_i
- If no condition is true then the conditional rule is undefined

Uses of Conditional Rules

- To define the value of a function
- $\min(x, y) = (x \leq y \Rightarrow x \mid x > y \Rightarrow y)$
- To define the meaning of a program
 - ▶ If $(x < y)$ then $z := x$ else $z := y$
 - ▶ $(x < y \Rightarrow z := x \mid x \geq y \Rightarrow z := y)$
 - ▶ $(x < y \Rightarrow x, y := x, y \mid x \geq y \Rightarrow x, y := y, x)$
- Conditional rules can be expressed in tables

Finite State Machines

- A FSM is a tuple $(S, s_0, I, O_E, O_O, T, E, C)$ where
- S is a finite set of states
- s_0 is the initial state in S ($s_0 \in S$)
- I is a finite set of inputs
- $T : S \times I \rightarrow S$ is the transition function
- O_E is a finite set of event outputs
- $E : S \times I \rightarrow O_E$ is the event output
- O_C is a finite set of condition outputs
- $C : S \rightarrow O_C$ is the condition output

Example 2D Data

- Problem Description
- Source Code

References I



David Gries and Fred B. Schneider.

A logical approach to discrete math.

Springer-Verlag Inc., New York, 1993.



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.