First Committee Meeting

Progress Report

Jason Balaci

McMaster University

Oct. 21st, 2021

Table of Contents

1. Introduction
2. Project
    - Drasil
    - Goal #1: Typed Expression Language
    - Goal #2: Model Discrimination – “ModelKinds”
3. References

Table of Contents

1. Introduction
2. Project
    - Drasil
    - Goal #1: Typed Expression Language
    - Goal #2: Model Discrimination – “ModelKinds”
3. References

Who am I?

Who am I?

- I am **Jason Balaci**

Me, Camping in Killarney Prov. Park, Fall 2019

Who am I?

- I am **Jason Balaci**
- Graduate of *McMaster University*, holding...

Me, Camping in Killarney Prov. Park, Fall 2019

Who am I?

- I am **Jason Balaci**
- Graduate of *McMaster University*, holding...
  - Hons. Actuarial and Financial Mathematics (B.Sc.)

Me, Camping in Killarney Prov. Park, Fall 2019

Who am I?

- I am **Jason Balaci**
- Graduate of *McMaster University*, holding...
    - Hons. Actuarial and Financial Mathematics (B.Sc.)
    - Minor in Computer Science

Me, Camping in Killarney Prov. Park, Fall 2019

Who am I?

- I am **Jason Balaci**
- Graduate of *McMaster University*, holding...
  - Hons. Actuarial and Financial Mathematics (B.Sc.)
  - Minor in Computer Science
- Currently pursuing a thesis-based Master's of Computer Science (M.Sc) at *McMaster University*, under the supervision of **Dr. Jacques Carette**.

Me, Camping in Killarney Prov. Park, Fall 2019

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
    - One (1) "Software" course

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) “Software” course
  - Either of:

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
    - One (1) "Software" course
    - Either of:
        - Two "Theory" courses, and one "Systems" course
        - One "Theory" course, and two "Systems" courses

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) "Software" course
  - Either of:
    - Two "Theory" courses, and one "Systems" course
    - One "Theory" course, and two "Systems" courses
- I've completed:

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
    - One (1) "Software" course
    - Either of:
        - Two "Theory" courses, and one "Systems" course
        - One "Theory" course, and two "Systems" courses
- I've completed:
    - CAS 701 "Logic & Discrete Mathematics" - Theory course, Fall 2020

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) "Software" course
  - Either of:
    - Two "Theory" courses, and one "Systems" course
    - One "Theory" course, and two "Systems" courses
- I've completed:
  - CAS 701 "Logic & Discrete Mathematics" - Theory course, Fall 2020
  - CAS 761 "Generative Programming" - Software course, Fall 2020

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) "Software" course
  - Either of:
    - Two "Theory" courses, and one "Systems" course
    - One "Theory" course, and two "Systems" courses
- I've completed:
  - CAS 701 "Logic & Discrete Mathematics" - Theory course, Fall 2020
  - CAS 761 "Generative Programming" - Software course, Fall 2020
  - CAS 763 "Certified Programming with Dependent Types" - Theory & Software course, Winter 2021

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) "Software" course
  - Either of:
    - Two "Theory" courses, and one "Systems" course
    - One "Theory" course, and two "Systems" courses
- I've completed:
  - CAS 701 "Logic & Discrete Mathematics" - Theory course, Fall 2020
  - CAS 761 "Generative Programming" - Software course, Fall 2020
  - CAS 763 "Certified Programming with Dependent Types" - Theory & Software course, Winter 2021
  - COMPSCI 6TB3 "Syntax-Based Tools and Compilers" - Systems course, Winter 2021

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Course-related progression

- I'm required to complete[1][2]:
  - One (1) "Software" course
  - Either of:
    - Two "Theory" courses, and one "Systems" course
    - One "Theory" course, and two "Systems" courses
- I've completed:
  - CAS 701 "Logic & Discrete Mathematics" - Theory course, Fall 2020
  - CAS 761 "Generative Programming" - Software course, Fall 2020
  - CAS 763 "Certified Programming with Dependent Types" - Theory & Software course, Winter 2021
  - COMPSCI 6TB3 "Syntax-Based Tools and Compilers" - Systems course, Winter 2021
- Together, the courses completed satisfies the "Courses Requirement" as mentioned in the academic calendar[1] and the "Regulations for the Computer Science M.Sc. Program" document[2].

[1] https://academiccalendars.romcmaster.ca/preview_program.php?catoid=45&poid=23470&returnto=9166

[2] http://www.cas.mcmaster.ca/cas/0files/reg_master_cs_2019a.pdf

Overview of Progression Towards C.S. M.Sc.

Thesis/research-related Progression

- Conducted “full-time” research for at least 1 full semester (Spring/Summer 2021), and “part-time” research during courses.

Overview of Progression Towards C.S. M.Sc.

Thesis/research-related Progression

- Conducted “full-time” research for at least 1 full semester (Spring/Summer 2021), and “part-time” research during courses.
- Continuing to research “full-time”.

Overview of Progression Towards C.S. M.Sc.

Thesis/research-related Progression

- Conducted “full-time” research for at least 1 full semester (Spring/Summer 2021), and “part-time” research during courses.
- Continuing to research “full-time”.
- Attended a thesis defence to learn about what to expect from a thesis defence (and learn about their research).

Overview of Progression Towards C.S. M.Sc.

Thesis/research-related Progression

- Conducted “full-time” research for at least 1 full semester (Spring/Summer 2021), and “part-time” research during courses.
- Continuing to research “full-time”.
- Attended a thesis defence to learn about what to expect from a thesis defence (and learn about their research).
- Supervisory committee is formed, and we are currently having our first supervisory committee meeting.
    - *Supervisor*: Dr. Jacques Carette
    - Dr. Spencer Smith
    - Dr. Wolfram Kahl

Table of Contents

1 Introduction

2 Project

- Drasil
- Goal #1: Typed Expression Language
- Goal #2: Model Discrimination – “ModelKinds”

3 References

Preface

What is Drasil?

Drasil...

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] `https://jacquescarette.github.io/Drasil/`

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.
    - Originally focused on scientific software (*Literate Scientific Software*).

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.
  - Originally focused on scientific software (*Literate Scientific Software*).
  - Focus expanded...

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.
    - Originally focused on scientific software (*Literate Scientific Software*).
    - Focus expanded...
- tries to “Generate All The Things”...

Drasil’s Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.
  - Originally focused on scientific software (*Literate Scientific Software*).
  - Focus expanded...
- tries to "Generate All The Things"...
  - with a focus on research software.

Drasil's Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Preface

What is Drasil?

Drasil...

- is managed by Dr. Carette & Dr. Smith.
- originates from the work of Dan Szymczak.
  - Originally focused on scientific software (*Literate Scientific Software*).
  - Focus expanded...
- tries to “Generate All The Things”...
  - with a focus on research software.
- has a website[1]!

Drasil’s Logo
[Carette et al., 2021][Yggdrasil - Wikipedia, 2021]

[1] https://jacquescarette.github.io/Drasil/

Drasil

“Generate All The Things!”

- TODO: here!

Drasil Case Studies

[1] https://jacquescarette.github.io/Drasil/#Sec:Examples

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
  - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
    - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
    - **Single Pendulum** - Observing the motion of a single pendulum.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
    - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
    - **Single Pendulum** - Observing the motion of a single pendulum.
    - **Double Pendulum** - Observing the motion of a double pendulum.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
  - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
  - **Single Pendulum** - Observing the motion of a single pendulum.
  - **Double Pendulum** - Observing the motion of a double pendulum.
  - **Game Physics** - Modelling of an open source 2D rigid body physics library used for games.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
  - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
  - **Single Pendulum** - Observing the motion of a single pendulum.
  - **Double Pendulum** - Observing the motion of a double pendulum.
  - **Game Physics** - Modelling of an open source 2D rigid body physics library used for games.
  - **Proportional Derivative Controller (PDController)** - Examining the output of a “Power Plant” (Process Variable) over time.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- Drasil currently contains a significant amount of Physics-related knowledge.
- As of writing, current case studies[1] are primarily related to physics, including:
  - **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
  - **Single Pendulum** - Observing the motion of a single pendulum.
  - **Double Pendulum** - Observing the motion of a double pendulum.
  - **Game Physics** - Modelling of an open source 2D rigid body physics library used for games.
  - **Proportional Derivative Controller (PDController)** - Examining the output of a “Power Plant” (Process Variable) over time.
  - **Solar Water Heating System (SWHS)** - Modelling of a solar water heating system with phase change material, predicting temperatures and change in heat energy of water and the PCM over time.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- *cont.d*[1]:
  - **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- *cont.d*[1]:
  - **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.
  - **Projectile** - Determining if a launched projectile hits a target, assuming no flight collisions.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

- *cont.d*[1]:
    - **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.
    - **Projectile** - Determining if a launched projectile hits a target, assuming no flight collisions.
    - **Slope Stability Analysis Program (SSP)** - Assessment of the safety of a slope (composed of rock and soil) subject to gravity, identifying the surface most likely to experience slip and an index of its relative stability (factor of safety).

[1] https://jacquescarette.github.io/Drasil/#Sec:Examples

Drasil Case Studies

- *cont.d*[1]:
  - **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.
  - **Projectile** - Determining if a launched projectile hits a target, assuming no flight collisions.
  - **Slope Stability Analysis Program (SSP)** - Assessment of the safety of a slope (composed of rock and soil) subject to gravity, identifying the surface most likely to experience slip and an index of its relative stability (factor of safety).
  - **Heat Transfer Coefficients between Fuel and Cladding in Fuel Rods (HGHC)** - Examining the heat transfer coefficients related to clad.

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Drasil Case Studies

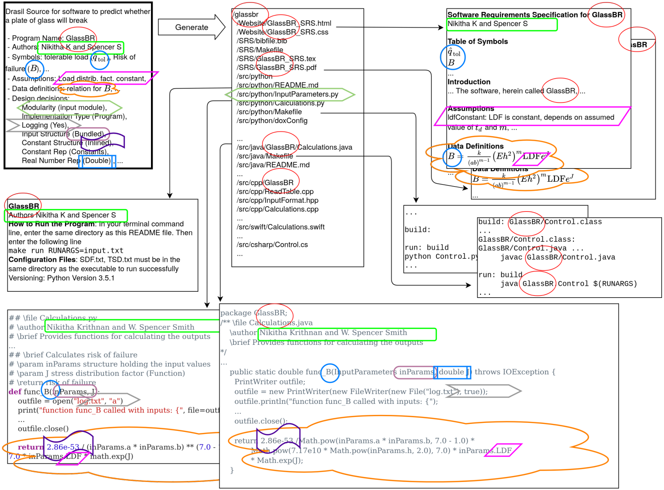
- *cont.d*[1]:
  - **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.
  - **Projectile** - Determining if a launched projectile hits a target, assuming no flight collisions.
  - **Slope Stability Analysis Program (SSP)** - Assessment of the safety of a slope (composed of rock and soil) subject to gravity, identifying the surface most likely to experience slip and an index of its relative stability (factor of safety).
  - **Heat Transfer Coefficients between Fuel and Cladding in Fuel Rods (HGHC)** - Examining the heat transfer coefficients related to clad.

The Drasil website is also generated by Drasil!

[1] `https://jacquescarette.github.io/Drasil/#Sec:Examples`

Taking a closer look at one of the examples: GlassBR

GlassBR Generates Code!



Drasil Source for software to predict whether a plate of glass will break

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{\text{tol}}$), Risk of failure (B), ...
- Assumptions: Load distrib. fact. constant,
- Data definitions: relation for B, ...
- Design decisions:
  Modularity (input module),
  Implementation Type (Program),
  Logging (Yes),
  Input Structure (Bundled),
  Constant Structure (Inlined),
  Constant Rep (Constants),
  Real Number Rep (Double), ...

```
/glassbr
 /Website/GlassBR_SRS.html
 /Website/GlassBR_SRS.css
 /SRS/bibfile.bib
 /SRS/Makefile
 /SRS/GlassBR_SRS.tex
 /SRS/GlassBR_SRS.pdf
 /src/python
 /src/python/README.md
 /src/python/InputParameters.py
 /src/python/Calculations.py
 /src/python/Makefile
 /src/python/doxConfig

 ...
 /src/java/GlassBR/Calculations.java
 /src/java/Makefile
 /src/java/README.md
 ...
 /src/cpp/GlassBR
 /src/cpp/ReadTable.cpp
 /src/cpp/InputFormat.hpp
 /src/cpp/Calculations.cpp
 ...
 /src/swift/Calculations.swift
 ...
 /src/csharp/Control.cs
 ...
```

Software Requirements Specification for GlassBR

Nikitha K and Spencer S

Table of Symbols

$\hat{q}_{\text{tol}}$

B

...

Introduction

... The software, herein called GlassBR, ...

Assumptions

ldfConstant: LDF is constant, depends on assumed value of t_d and m, ...

Data Definitions

$B = \frac{k}{(ab)^{m-1}}\left(Eh^2\right)^m \text{LDF} e^J$

...

sBR

Data Definitions

$B = \frac{k}{(ab)^{m-1}}\left(Eh^2\right)^m \text{LDF} e^J$

...

GlassBR

Authors Nikitha K and Spencer S

How to Run the Program: In your terminal command line, enter the same directory as this README file. Then enter the following line

`make run RUNARGS=input.txt`

Configuration Files: SDF.txt, TSD.txt must be in the same directory as the executable to run successfully

Versioning: Python Version 3.5.1

```
...

build:

run: build
python Control.py
...
```

```
build: GlassBR/Control.class
...
GlassBR/Control.class:
GlassBR/Control.java ...
    javac GlassBR/Control.java

run: build
    java GlassBR.Control $(RUNARGS)
...
```

```
## \file Calculations.py
# \author Nikitha Krithnan and W. Spencer Smith
# \brief Provides functions for calculating the outputs
...
## \brief Calculates risk of failure
# \param inParams structure holding the input values
# \param J stress distribution factor (Function)
# \return risk of failure
def func_B(inParams, J):
    outfile = open("log.txt", "a")
    print("function func_B called with inputs: {", file=outf
    ...
    outfile.close()

    return 2.86e-53 / (inParams.a * inParams.b) ** (7.0 -
7.0 * inParams.LDF * math.exp(J)
```

```
package GlassBR;
/** \file Calculations.java
  \author Nikitha Krithnan and W. Spencer Smith
  \brief Provides functions for calculating the outputs
*/
...
  public static double func_B(InputParameters inParams, double J) throws IOException {
    PrintWriter outfile;
    outfile = new PrintWriter(new FileWriter(new File("log.txt", true));
    outfile.println("function func_B called with inputs: {");
    ...
    outfile.close();

    return 2.86e-53 / Math.pow(inParams.a * inParams.b, 7.0 - 1.0) *
          Math.pow(7.17e10 * Math.pow(inParams.h, 2.0), 7.0) * inParams.LDF
          * Math.exp(J);
  }
```

Knowledge flow from “knowledge-base”/source to artifacts, by Dr. Spencer Smith

Which case studies currently generate code?

- **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.

Which case studies currently generate code?

- **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
- **Proportional Derivative Controller (PDController)** - Examining the output of a “Power Plant” (Process Variable) over time.

Which case studies currently generate code?

- **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
- **Proportional Derivative Controller (PDController)** - Examining the output of a “Power Plant” (Process Variable) over time.
- **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.

Which case studies currently generate code?

- **GlassBR** - Predicting whether or not a glass slab is likely to resist a specified blast.
- **Proportional Derivative Controller (PDController)** - Examining the output of a “Power Plant” (Process Variable) over time.
- **SWHS without Phase Change Material (NoPCM)** - Modelling of a solar water heating system without phase change material, predicting temperatures and change in heat energy of water and the PCM over time.
- **Projectile** - Determining if a launched projectile hits a target, assuming no flight collisions.

Why don't all case studies generate software artifacts?

Where will I be contributing?

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!
- The SRS documents are generated!

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!
- The SRS documents are generated!

Generating view-only data (e.g., SRS documents) is considerably easier than generating working code.

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!
- The SRS documents are generated!

Generating view-only data (e.g., SRS documents) is considerably easier than generating working code.

A few, notable, blocking problems:

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!
- The SRS documents are generated!

Generating view-only data (e.g., SRS documents) is considerably easier than generating working code.

A few, notable, blocking problems:

- Confidently generating usable software artifacts without strong type information places significant stress on developers, resulting in a higher likelihood of bugs in artifacts.

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Why don't all case studies generate software artifacts?

Where will I be contributing?

After all,

- They're all covered under "well-understood" domains!
- The SRS documents are generated!

Generating view-only data (e.g., SRS documents) is considerably easier than generating working code.

A few, notable, blocking problems:

- Confidently generating usable software artifacts without strong type information places significant stress on developers, resulting in a higher likelihood of bugs in artifacts.
- Existing "theories"/"*Models"[1] don't expose enough information. They must be enriched, so that we can pull more information from them in straightforward manner.

[1] Terminology is currently being changed, but is not reflected in many documents yet.

Goal #1: Typed Expression Language

The Problem

Goal #1: Typed Expression Language

The Problem

- Ensure only admissible expressions are used in GOOL-supported languages, and that all expressions are coherent.

Goal #1: Typed Expression Language

The Problem

- Ensure only admissible expressions are used in GOOL-supported languages, and that all expressions are coherent.
- Eases developer cognitive load when writing expressions, as they will need to ensure their expressions are coherent, or else a type error will be thrown.

Goal #1: Typed Expression Language

What makes up a “good” solution?

Goal #1: Typed Expression Language

What makes up a "good" solution?

- Catches, within reason, all possible scenarios where an expression goes awry.

Goal #1: Typed Expression Language

What makes up a "good" solution?

- Catches, within reason, all possible scenarios where an expression goes awry.
- Allows GOOL code generator to also become typed!

Goal #1: Typed Expression Language

Current Progression

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)
    - ‘Expr’ is a discrete, directly computable language (calculator possible)

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)
    - 'Expr' is a discrete, directly computable language (calculator possible)
    - Created 'ModelExpr' contains all possible expressions we would want to write on pencil-and-paper. There are still a few operations left in "Expr" that need to be moved over, however.

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)
    - ‘Expr’ is a discrete, directly computable language (calculator possible)
    - Created ‘ModelExpr’ contains all possible expressions we would want to write on pencil-and-paper. There are still a few operations left in “Expr” that need to be moved over, however.
        - Theories that rely on discussion of terms only found ‘ModelExpr’ may only have code generation made possible if we have rich enough data (see goal #2)

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)
    - ‘Expr’ is a discrete, directly computable language (calculator possible)
    - Created ‘ModelExpr’ contains all possible expressions we would want to write on pencil-and-paper. There are still a few operations left in “Expr” that need to be moved over, however.
        - Theories that rely on discussion of terms only found ‘ModelExpr’ may only have code generation made possible if we have rich enough data (see goal #2)
    - ‘CodeExpr’ contains a total conversion from ‘Expr’ with a few extra functionalities for GOOL

Goal #1: Typed Expression Language

Current Progression

- Split core mathematical expression language (Expr) into 3 variants (Expr, ModelExpr, and CodeExpr)
  - ‘Expr’ is a discrete, directly computable language (calculator possible)
  - Created ‘ModelExpr’ contains all possible expressions we would want to write on pencil-and-paper. There are still a few operations left in “Expr” that need to be moved over, however.
    - Theories that rely on discussion of terms only found ‘ModelExpr’ may only have code generation made possible if we have rich enough data (see goal #2)
  - ‘CodeExpr’ contains a total conversion from ‘Expr’ with a few extra functionalities for GOOL
  - Created a “typed tagless final” smart constructor encoding for writing expressions in “Expr” (or, optionally, ModelExpr).

Goal #1: Typed Expression Language

What are the next steps?

Goal #1: Typed Expression Language

What are the next steps?

- Continuing to remove terms.

Goal #1: Typed Expression Language

What are the next steps?

- Continuing to remove terms.
- Moving literals from “Expr” & “ModelExpr” into their own small language, so that areas that want *strictly* literals can also have stronger restrictions on allowed data (terms).

Goal #2: Model Discrimination – “ModelKinds”

The Problem

Goal #2: Model Discrimination – “ModelKinds”

The Problem

- “RelationConcept”s don’t contain enough information on their own to be a core component usable in code generation.

Goal #2: Model Discrimination – “ModelKinds”

The Problem

- “RelationConcept”s don’t contain enough information on their own to be a core component usable in code generation.
- They were essentially “Relation”s (“Expr”s) with a natural language description of them.

Goal #2: Model Discrimination – “ModelKinds”

The Problem

- “RelationConcept”s don’t contain enough information on their own to be a core component usable in code generation.
- They were essentially “Relation”s (“Expr”s) with a natural language description of them.
- If the “shape” of the expressions are not uniform, then writing more “interpreters”/“views”/code generators for them required difficult pattern analysis. It’s also not a total-conversion.

Goal #2: Model Discrimination – “ModelKinds”

What makes up a “good” solution?

Goal #2: Model Discrimination – “ModelKinds”

What makes up a “good” solution?

- A good solution involves making the “Relation”s a “view” of a more data-rich specialized container for each kind of “*Model”.

Goal #2: Model Discrimination – “ModelKinds”

What makes up a “good” solution?

- A good solution involves making the “Relation”s a “view” of a more data-rich specialized container for each kind of “*Model”.
- “ModelKinds”

Goal #2: Model Discrimination – “ModelKinds”

What makes up a “good” solution?

- A good solution involves making the “Relation”s a “view” of a more data-rich specialized container for each kind of “*Model”.
- “ModelKinds”
- By constructing our final data views through “more steps” (e.g., with more depth), we create a system of specialization, and we can start to see how specialization will occur.

Goal #2: Model Discrimination – "ModelKinds"

Current Progression

Goal #2: Model Discrimination – “ModelKinds”

Current Progression

- Considerable number of “theories”/“*Models” have been restructured, but there are still many that are pending typing. Most are best to be done once we have a typed expression language, and the rest are differential equation-related models (primarily Dong’s domain).

Goal #2: Model Discrimination – “ModelKinds”

What are the next steps?

Goal #2: Model Discrimination – “ModelKinds”

What are the next steps?

- Understanding what kinds of needs we have for “collections”, pushing this information back into the typed expression language (once that is fully typed), and then further creating model containers for these models.

Goal #2: Model Discrimination – “ModelKinds”

What are the next steps?

- Understanding what kinds of needs we have for “collections”, pushing this information back into the typed expression language (once that is fully typed), and then further creating model containers for these models.
- For the differential equation-related models, we will need to build appropriate models for each kind.

Acknowledgements

Fin.

Thank you!

Table of Contents

1 Introduction

2 Project

- Drasil
- Goal #1: Typed Expression Language
- Goal #2: Model Discrimination – “ModelKinds”

3 References

References I

- Carette, J., Smith, S., Balaci, J., Hunt, A., Wu, T.-Y., Crawford, S., Chen, D., Szymczak, D., MacLachlan, B., Scime, D., and Niazi, M. (2021).
  Drasil.

- Yggdrasil - Wikipedia (2021).
  Yggdrasil.