User Manual for the Physics Derivation Graph

Ben Payne[1]*, Michael Goff[2]
[1]*Department of Fun, University Name & Town, city, State Zip*
[2]*Department of Mathematics, University of Maryland, Baltimore 21228*

September 8, 2015

Abstract

Overview of current implementation. It is assumed here that the Introduction to the Physics Derivation Graph has been read.

1 Introduction

The Physics Derivation Graph is a project designed to document mathematical physics knowledge. The mathematics necessary to describe physics can be written down, in this case in the form of a graph. The core of this project is a set of databases storing that content and an associated set of scripts for various manipulations of the content. Those scripts include the ability to generate visualizations of the graph, and to generate reports based on the graph content.

Section 2 describes the databases which store these values.

2 Databases

The content associated with physics knowledge takes the form of expressions (*equations and inequalities*), symbols, and inference rules. *An inference rule is an atomic transformation of one expression to another.* An example is provided here.

Frequency f and period T are related by

$$T\ f = 1 \tag{1}$$

Thus, frequency in terms of the period is

$$f = 1/T \tag{2}$$

The relation between Eq. (1) and Eq. (2) is that both sides of Eq. (1) were divided by T.

In this example, there are two mathematical expressions: Eq. (1) and Eq. (2). These expressions are related by an inference rule: "Divide both sides of first equation by a value to yield the second equation." This inference rule takes an argument, referred to here as the "feed", which in this example is T. This set of steps is shown graphically in Fig. **??**. This simple step between two expressions is documented in five separate databases: expressions, symbols, feeds, inference rules, and connections.

Each expression, symbol, and inference rule appears only once in the respective database. Each time an expression, symbol, or inference rule is used in a derivation, that unique instance is referenced. This referencing of unique expressions, symbols, and inference rules is done using a numeric identifier (alphanumeric for inference rules).

In the expressions database, each expression has an associated 10 digit number in base 10. For example, Eq. (2) is 2113211456 and Eq. (1) is 2131616531. Similarly, the symbols have unique 15 digit numeric identfiers. Period T is 192938440120938 and frequency f is 102938475990112. The database of expressions references these symbol identifiers – expression 2113211456 uses symbols 192938440120938 and 102938475990112.

The connections database is the set of derivations. A derivation involving Eq. (2) refers to the expression identifier 2113211456 and relates it to 2131616531 by an inference rule which is identified by the alphanumeric string "dividebothsidesby". This inference rule takes an argument, referred to here as a "feed." The feed database contains the value, in this example T and a 7 digit identifier 8837284. Feeds are not unique, though the identifiers used are. The T referred to for this use of the inference rule "dividebothsidesby" does not need to be associated with other instances

*Corresponding author: ben.is.located@gmail.com

```
"frequency relations",6, "expression",8482459,3131111133,"infrule",2939482,subRHSofEqXintoEqY
```

Figure 1: Fig. 1: connections database content example.

of "dividebothsidesby".

The Physics Derivation Graph is designed to show one instance of each expression, but feeds and inference rules will have duplicates on the graph. To allow for this, the connections database has a unique 7 digit numeric identifier assocaited with each use of an inference rule.

The databases supporting the Physics Derivation Graph are implemented using comma-separated (CSV) plain-text files with no headers.

2.1 Implementation: Symbols

```
<symbol>
  <symbol_name></symbol_name>
  <symbol_punid></symbol_punid>
  <type></type>
  <description></description>
  <cas_sympy></cas_sympy>
  <units></units>
</symbol>
```

2.2 Implementation: Expressions

```
3131111133,"T=1/f"
```

2.3 Implementation: Connections

In the connections database, the

3 Database manipulation

The databases are manipulated using software written in Python[1] version 2.7.6. Each script in the `bin` directory performs a single task. Common functions are stored in `lib/lib_physics_graph.py`. The scripts in `bin` are described here in order of increasing complexity.

3.1 Listing

The two simplest scripts are for listing content from the connections and inference rules databases, respectively.

```
bin/list_connection_sets.py
bin/list_inference_rules.py
```

The output of listing the connection sets are the derivations which are listed in the connections database.

```
bin/list_connection_sets.py
    frequency_relations
    quadratic_equation_derivation
    euler_equation_proof
    euler_equation_trig_sqrt
    example_of_Godel_problem
    particle_in_a_1D_box
    ...
```

3.2 Statistical Analysis

There are three scripts which build on the listing functionality and count instances of inference rules, expressions, and symbols, respectively.

```
popularity_of_inference_rules.py
popularity_of_statements.py
popularity_of_symbols.py
```

```
bin/popularity_of_statements.py
    10 4938429483 \exp(i\ x) = \cos(x)+...
    6 3948574230 \psi(\vec{r},t) = ...
    5 5727578862 \frac{d^2}{dx^2} ...
    ...
```

For example, Eq. (2) is referenced twice as of this writting.

3.3 Utilities

The utility scripts are meant for use by developers of the content.

```
create_pdf_of_statements.py
create_pictures_of_inference_rules.py
create_pictures_of_statements_and_feeds.py
generate_new_random_index.py
```

Creating a PDF of the expressions can be used to validate the LaTeXsyntax of content in `databases/expressions_database.csv`

3.4 Primary functions

```
check_connections_using_CAS_sympy.py
build_connection_set_pdf.py
build_connections_graph.py
```

The `build_*.py` scripts are described in detail in sections 4 and 5, respectively.

Checking the connections using SymPy is not fully supported.

4 Report generation

LaTeX[2] PDFs can be created by running `bin/build_connection_set_pdf.py`

5 Visualization of Graph

DOT language[3]

`bin/build_connections_graph.py`

6 Creating a new Derivation

6.1 New Symbol

First, pick a symbol from your derivation.

Second, check whether that symbol exists in the symbol database. If it does, you are done.

If your symbol is not already in the symbol database, the third step is to get a new random index ("available symbol numeric identifier").

Copy the symbol CSV fields to create a new entry

Place the symbol and new numeric identifier into the appropriate field

6.2 New expression

First, pick the expression to be added.

Second, check whether that expression exists in the expression database. If it does, you are done

If your expression is not already in the expression database, the third step is to get a new random index ("available expression numeric identifier").

In the expressions database, copy the expression CSV fields into the appropriate derivation.

Place the LaTeXexpression and numeric identifier in the appropriate field.

Determine the SymPy equivalent. Include that in the CSV

Determine the symbols in the expression. See section 6.1. Include the symbol numeric identifiers in the expression database.

6.3 New Connection

An inference rule connects expressions

6.4 New Derivation

A derivation is composed of a sequence of expressions connected by inference rules

7 Summary

8 Bibliography

References

[1] Python language reference, version 2.7, 2015.

[2] Leslie Lamport. *LaTeX: A Document Preparation System*, second edition, 1994.

[3] Graphviz, 2015.