R.C. Development

User Interface Design Document

CLIF Parser, designed for Torsten Hahmann and Jake Emerson

Designed by Reading Club Development:
Matthew Brown, Gunnar Eastman, Jesiah Harris , Shea Keegan, Eli Story

Version 1.0
Nov. 29, 2022

Table of Contents

1. Introduction

In this section of this User Interface Design Document (UIDD), the project will be introduced along with the purpose of the UIDD, the references used and compiled by Reading Club Development (RCD), the purpose of the product, and the scope of the product.

This is a capstone project for Dr. Torsten Hahmann and Jake Emerson, in partial fulfillment of the Computer Science BS degree for the University of Maine. This UIDD will detail the user interface that will be produced to interact with the product and document the consent of the team members and the client.

1.1 Purpose of This Document

This UIDD is meant to expand upon the user interface details for the Common Logic Interchange Format (CLIF) Parser that RCD has been tasked with developing. The intended readership of this document consists of the client and the RCD team so as to effectively develop the proposed CLIF Parser.

1.2 References

Macleod, Dr. Torsten Hahmann, GitHub, 2022, https://github.com/thahmann/macleod.

SDD, Reading Club Development, 2022. https://docs.google.com/document/d/1H77pVpVR7mhu8ciDFAjY1WjnKQC3SNy9Po1MRIDwRcE/edit

SRS, Reading Club Development, 2022. https://docs.google.com/document/d/1vr3CD5g3azf6OBhu2w5QEy9lcAv1LxDzvFqTvZ7-CwI/edit

2. User Interface Standards

This section will outline the standards used when developing the user interface, to ensure consistency. It will describe what information will be available to the user, in which windows, and how users will interact with the system. This section will also outline how errors will be presented to the user.

The user interface will be composed of 5 sections: file navigation, file editor, console, error list, and toolbar. Each section will be available to the user at all times while using the program even if a certain section is not being utilized at a given time.

2.1 File Navigation

On the left side of the screen there will be a pane the height of the program window that will contain the file navigation hierarchy. The user will be able to navigate through the hierarchy by clicking a folder to see its contents. Clicking on a file will open that file in the file editor window so long as the file type is supported. Consistency checks will be run on a file as it is being opened.

2.2 File Editor

In the middle of the program window, there will be a pane that will display a file that was chosen from the file navigation pane. In this editor pane, the user will be able to make edits to the file directly. If Macleod encounters a syntactical error while parsing a CLIF file, it will open that file in this pane so that the error may be fixed.

2.3 Console

At the bottom of the program window, there will be a pane that will contain a console. This console will be a place where the logs that may be printed during the runtime of a program will be displayed.

2.4 Error List

On the left side of the program window there will be a pane that contains a list of parsing errors that the program occurred during run time. This list will allow the user to quickly navigate to the location in the file where the error occurred. Each error will list the line it was encountered on and any other information Macleod was able to discern.

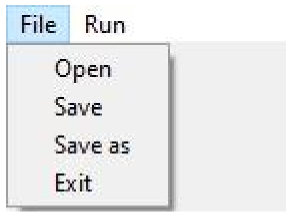
2.5 Toolbar

At the top of the program window, running the length of the program window, there will be a toolbar. In this toolbar the user will be able to select the language they would like the file currently open in the file editing pane to be parsed in. There will also be the option to run the parser on said file. In the toolbar there will be a button which will allow the user to save the edits made to the file that is actively open in the file editor. There will also be a button present to run consistency checks on a currently open file. The toolbar will also be a place where the other actions will be placed as development continues and user actions are created.

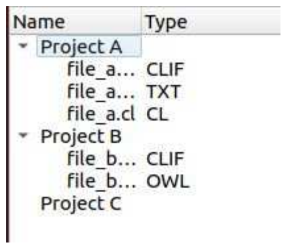
3. User Interface Walkthrough

Present in this section is a navigation diagram, illustrating how, from beginning to end, a user would translate a CLIF file using the system's user interface. Afterwards, all other windows will be illustrated and described.

3.1 Navigation Diagram

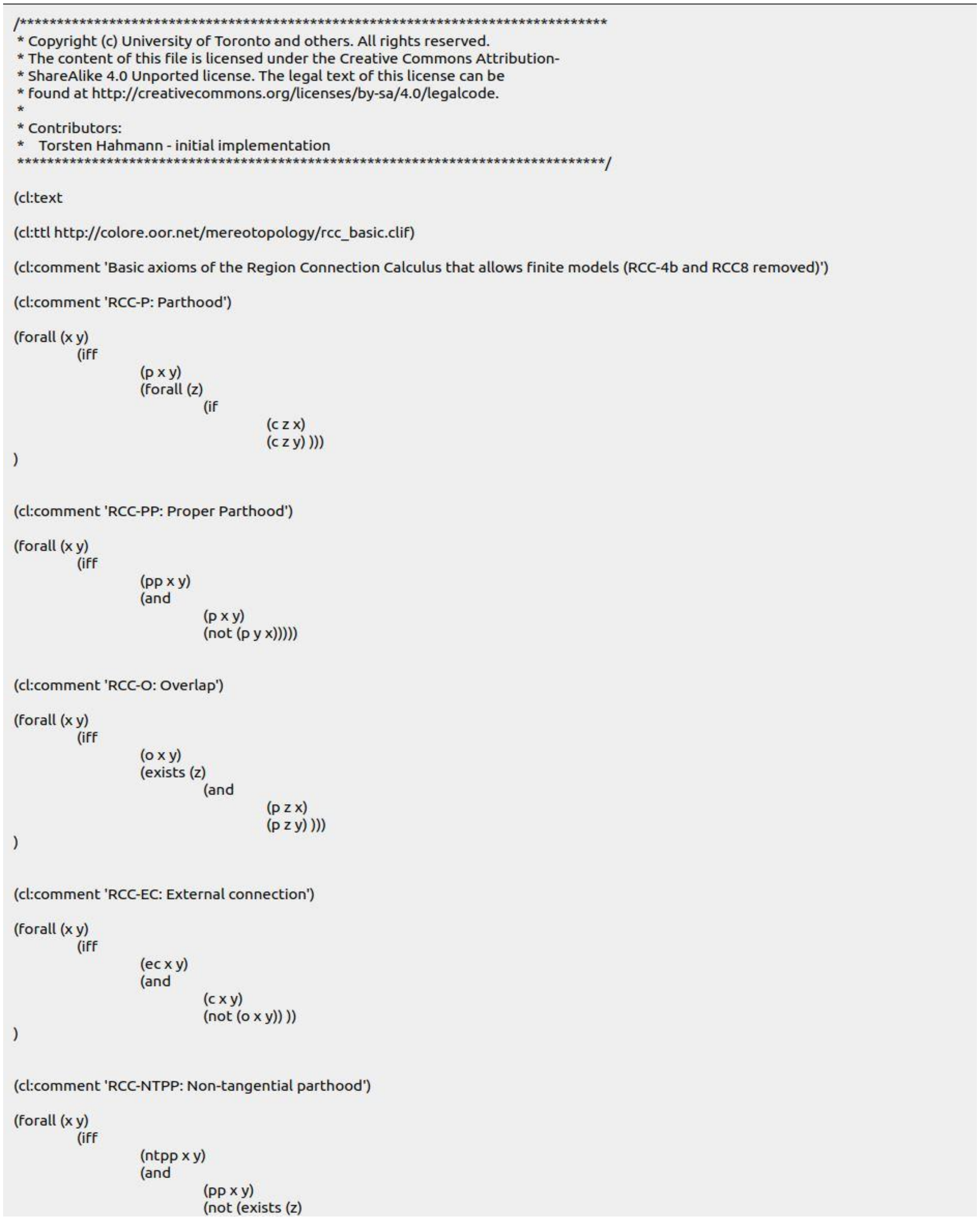


Users will be able to begin a project and open a CLIF file into the editor through the "File" tool in the toolbar. Once a project has been opened, the project will be displayed in a tree-like structure in the project window on the left of the editor.



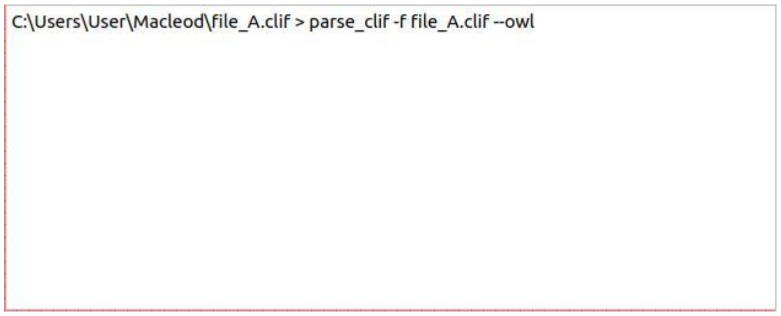
Name	Type
Project A	
file_a...	CLIF
file_a...	TXT
file_a.cl	CL
Project B	
file_b...	CLIF
file_b...	OWL
Project C	

CLIF files that have been opened by a user will be available for editing and debugging in the text editor as is pictured here:



```
/*********************************************************************************
 * Copyright (c) University of Toronto and others. All rights reserved.
 * The content of this file is licensed under the Creative Commons Attribution-
 * ShareAlike 4.0 Unported license. The legal text of this license can be
 * found at http://creativecommons.org/licenses/by-sa/4.0/legalcode.
 *
 * Contributors:
 *    Torsten Hahmann - initial implementation
 *********************************************************************************/

(cl:text

(cl:ttl http://colore.oor.net/mereotopology/rcc_basic.clif)

(cl:comment 'Basic axioms of the Region Connection Calculus that allows finite models (RCC-4b and RCC8 removed)')

(cl:comment 'RCC-P: Parthood')

(forall (x y)
		(iff
				(p x y)
				(forall (z)
						(if
								(c z x)
								(c z y) )))
)


(cl:comment 'RCC-PP: Proper Parthood')

(forall (x y)
		(iff
				(pp x y)
				(and
						(p x y)
						(not (p y x)))))


(cl:comment 'RCC-O: Overlap')

(forall (x y)
		(iff
				(o x y)
				(exists (z)
						(and
								(p z x)
								(p z y) )))
)


(cl:comment 'RCC-EC: External connection')

(forall (x y)
		(iff
				(ec x y)
				(and
						(c x y)
						(not (o x y)) ))
)


(cl:comment 'RCC-NTPP: Non-tangential parthood')

(forall (x y)
		(iff
				(ntpp x y)
				(and
						(pp x y)
						(not (exists (z)
```

Users will be able to run various commands such as parse, translate, etc through both the "Run" function on the toolbar (to be expanded upon later in development) as well as by running Macleod scripts/commands in the console.

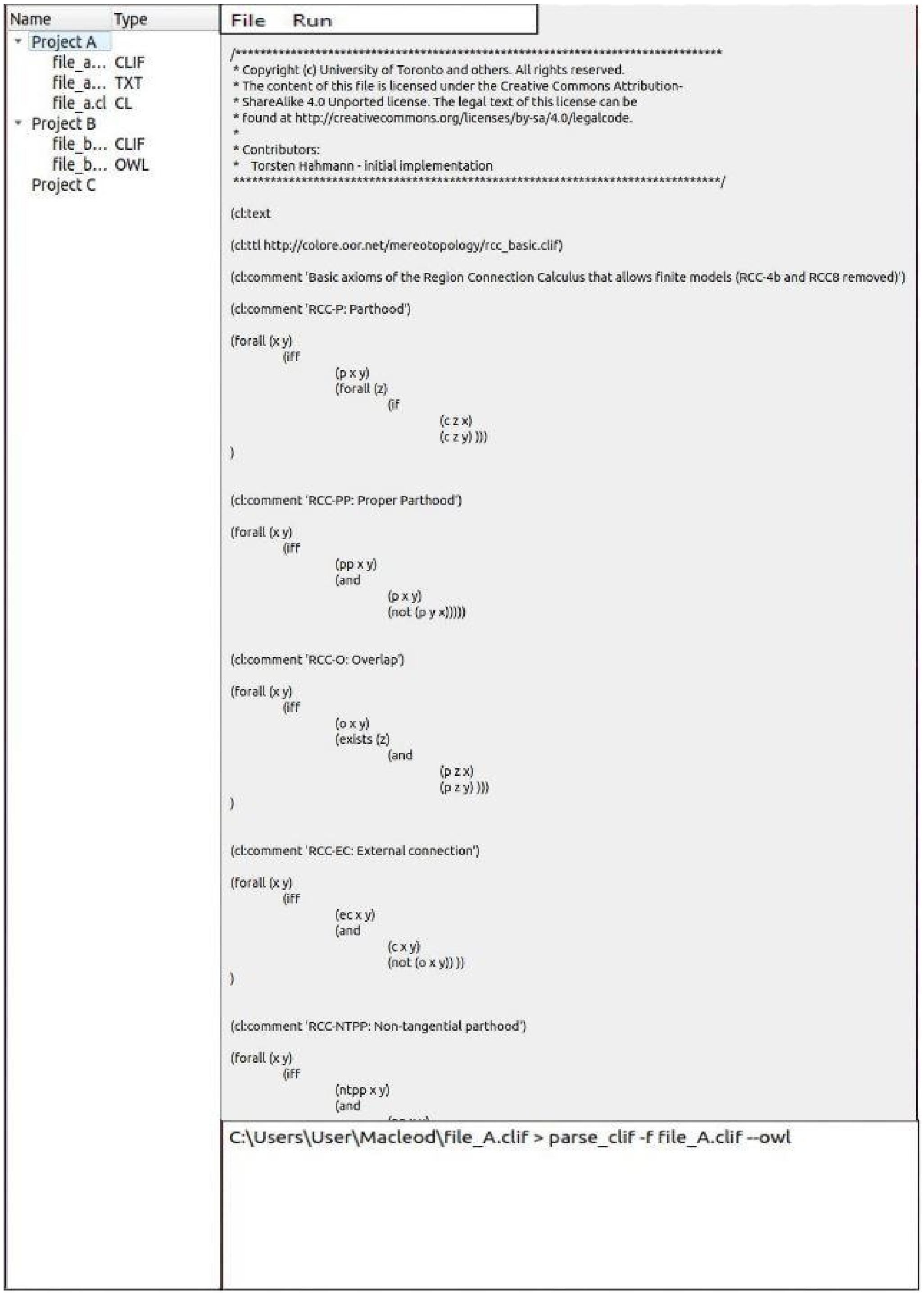


```
C:\Users\User\Macleod\file_A.clif > parse_clif -f file_A.clif --owl
```

The user interface will have the option for users to use the console to run scripts and view output, and a second tab will be available to view an error list that contains a list of parsing errors that the program occurred during run time.

3.2 Remaining Windows

An example of the combined user interface can be seen below:



Name	Type
Project A	
file_a…	CLIF
file_a…	TXT
file_a.cl	CL
Project B	
file_b…	CLIF
file_b…	OWL
Project C	

```
/*********************************************************************************
 * Copyright (c) University of Toronto and others. All rights reserved.
 * The content of this file is licensed under the Creative Commons Attribution-
 * ShareAlike 4.0 Unported license. The legal text of this license can be
 * found at http://creativecommons.org/licenses/by-sa/4.0/legalcode.
 *
 * Contributors:
 *    Torsten Hahmann - initial implementation
 *********************************************************************************/

(cl:text

(cl:ttl http://colore.oor.net/mereotopology/rcc_basic.clif)

(cl:comment 'Basic axioms of the Region Connection Calculus that allows finite models (RCC-4b and RCC8 removed)')

(cl:comment 'RCC-P: Parthood')

(forall (x y)
        (iff
                (p x y)
                (forall (z)
                        (if
                                (c z x)
                                (c z y) )))
)


(cl:comment 'RCC-PP: Proper Parthood')

(forall (x y)
        (iff
                (pp x y)
                (and
                        (p x y)
                        (not (p y x)))))


(cl:comment 'RCC-O: Overlap')

(forall (x y)
        (iff
                (o x y)
                (exists (z)
                        (and
                                (p z x)
                                (p z y) )))
)


(cl:comment 'RCC-EC: External connection')

(forall (x y)
        (iff
                (ec x y)
                (and
                        (c x y)
                        (not (o x y)) ))
)


(cl:comment 'RCC-NTPP: Non-tangential parthood')

(forall (x y)
        (iff
                (ntpp x y)
                (and
```

```
C:\Users\User\Macleod\File_A.clif > parse_clif -f File_A.clif --owl
```

4. Data Validation

This section will detail what types of data Macleod will and will not accept as input.

4.1 Input via GUI

The File Editor will be able to open text-like files, including .clif, .cl, and .txt.

When using the GUI to parse a file, the GUI will request a filepath, which will be a plaintext string that must end in either a slash, to indicate a folder, or ".clif", to indicate an individual file. In the case of a CLIF file, Macleod will parse that file and all files it imports, though in the case of a folder, Macleod will parse all CLIF files in that folder.

The GUI will ask for an output filename, though if none is provided, it shall make its own based on the input filename. The output filename must be a string that does not end in a slash or a file extension, as Macleod will automatically add the file extension to the end of the filename. If no output name is provided, Macleod will create a folder called "conversions" in the same folder as the input and put the output into the conversions folder.

The GUI will also provide selectable options to translate the file into TPTP, OWL, LaTeX, and LADR formats. These options will not be mutually exclusive, should a user wish to translate a file to all of the previously listed formats.

Finally, the GUI will provide an option to include axioms in the import closure of the CLIF file. This will simply be a checkbox which will be checked by default where, if checked, the parsing will be performed as though the –resolve parameter was included. The user may uncheck it if they wish to not include the –resolve parameter.

4.2 Input via Terminal

Given that using Macleod via the terminal mostly works at the time of writing, it will remain largely unchanged.

The parse_clif command will still be the main function used to translate clif files, and will require only a filename, either ending in .clif to signify a single CLIF file, or in a slash, to signify a folder. Ideally we will be able to remove the -f parameter and be able to detect the .clif at the end of the filename to identify an input of one file.

Optional parameters for this command will be a language parameter: either –tptp, –owl, or –ladr, to signify the format to translate the CLIF file into. If the language parameter is not provided, Macleod will simply parse and print the CLIF file as it currently does.

The final optional parameter will be the –resolve parameter, whose functionality will remain unchanged.

Appendix C

This appendix will outline the approximate contributions of each of the team members of RCD to the completion of this SDD.

Matthew Brown

- Conducted sister team review
- Edited and formatted document
- Contributed 20% of the document

Gunnar Eastman

- Wrote Section 1: Introduction
- Wrote Section 4: Data Validation.
- Wrote the introduction for Section 2: User Interface Standards, edited the rest of the section.
- Contributed 30% of the document

Jesiah Harris

- Wrote the User Interface Walkthrough
- Contributed 22.5% of the document

Shea Keegan

- Wrote the User Interface Standards
- Contributed 5% of the document

Eli Story

- Aided in management
- Wrote Section 2
- Contributed 22.5% of the document

Appendix A

This appendix details the expectations that RCD shall uphold to the client upon completion of this document, and how future changes to this document shall be made.

RCD and the client, upon the signing of the document, are agreeing that this UIDD contains a compilation of the user interface (UI) necessary for the CLIF Parser. RCD and the client agree that this UI is to be developed over the course of the Fall 2022 and Spring 2023 University of Maine semesters. The team, RCD, agrees that this UI is meant to be agile and flexible in nature, so if the need arises, the requirements may change in accordance with the client's wishes.

Any changes made to this document must be approved by all members of RCD and the client via signatures to an additional appendix wherein the changes are enumerated and detailed. Changes to this document include, but are not limited to, the shifting of architectural design as to be in accordance with the Capstone requirements for the University of Maine course this project is managed through. The signing of this appendix consents all members of RCD and the client that this structure of implementing changes is acceptable.

Name:	Signature:	Date:
Torsten Hahmann	[illegible]	11/29/22
Jake Emerson	R Jacob Emerson	11/29/22
Matthew Brown	Matthew Brown	11/29/22
Gunnar Eastman	Gunnar Eastman	11/29/22
Jesiah Harris	JH	11/29/22
Shea Keegan	[illegible]	11/29/22
Eli Story	Elijah Story	11/29/22

Customer Comments:

Appendix B

This appendix will contain the agreement that all members of RCD have read and consent to the document in its entirety.

Through signing this appendix, we, as the members of RCD, agree that we have reviewed this document fully, we agree to the formatting of this document, and agree to the content that is located within this UIDD. Each member of RCD may have minor disagreements with certain parts of this document, though by signing below, we agree that there are not any major points of contention within this SRS. We have all agreed to these terms and placed our signatures below.

Name:	Signature:	Date:
Matthew Brown	[illegible]	11/29/22
Comments:		
Gunnar Eastman	Gunnar Eastman	11/29/22
Comments:		
Jesiah Harris	JH	11/29/22
Comments:		
Shea Keegan	[illegible]	11/29/22
Comments:		
Eli Story	Elijah Story	11/29/22
Comments:		