CSNePS USER'S MANUAL[1]

Stuart C. Shapiro
and
The SNePS Implementation Group

Department of Computer Science and Engineering
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260-2000

August 18, 2014

[1]The development of SNePS was supported in part by: the National Science Foundation under Grants IRI-8610517 and REC-0106338; the Defense Advanced Research Projects Agency under Contract F30602-87-C-0136 (monitored by the Rome Air Development Center) to the Calspan-UB Research Center; the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB DC 20332 under Contract No. F30602-85-C-0008, which supported the Northeast Artificial Intelligence Consortium (NAIC); NASA under contract NAS 9-19335 to Amherst Systems, Inc.; ONR under contract N00014-98-C-0062 to Apple Aid, Inc.; the U.S. Army CECOM Intelligence and Information Warfare Directorate (I2WD) through a contract with CACI Technologies and through Contract #DAAB-07-01-D-G001 with Booze·Allen & Hamilton; and a Multidisciplinary University Research Initiative (MURI) grant (Number W911NF-09-1-0392) for "Unified Research on Network-based Hard/Soft Information Fusion," issued by the US Army Research Office (ARO) under the program management of Dr. John Lavery.

Over the years, many people have contributed to the design and implementation of SNePS, and to the writing of successive versions of the SNePS User's Manual. They constitute "The SNePS Implementation Group" cited on the title page, and I am grateful to them. They are listed here. If I have inadvertently omitted anyone's name, or have mispelled anyone's name, please let me know, and I will correct it for the next printing of this Manual.

Syed S. Ali
Michael J. Almeida
Charles W. Arnold
Robert J. Bechtel
Sudhaka Bharadwaj
Bharat Bhushan
Jonathan Bona
Jong S. Byoun
Alistair E. Campbell
Scott S. Campbell
Hans Chalupsky
Chung M. Chan
Robert G. Church
Joongmin Choi
Chi C. Choy
Soon Ae Chun
Maria R. Cravo
Zuzana Dobes
Gerard F. Donlon
Nicholas E. Eastridge
Elissa Feit
David Forster
Richard B. Fritzson
James Geller
Susan M. Haller
Richard G. Hull
Haythem Ismail
Frances L. Johnson
Steven D. Johnson
Darrel L. Joy
Sudha Kailar
Michael Kandefer
Sijun Kang
Deepak Kumar
Stanley C. Kwasny
John S. Lewocz
Naicong Li
John D. Lowrance
Christopher Lusardi
Anthony S. Maida
Mark D. Malamut
Nuno Mamede
João P. Martins
Pedro A. Matos
Donald P. McKay
James P. McKew
Ernesto J. Morgado
William A. Neagle
Jeannette G. Neal
Jane Terry Nutter
Rafail Ostrovsky
Sandra L. Peters
Anthony S. Petre
David R. Pierce
Carlos Pinto-Ferreira
William J. Rapaport
Victor H. Saks
John F. Santore
Daniel R. Schlegel
A. Patrice Seyed
Harold L. Shubin
Reid G. Simmons
Benjamin R. Spigle, Jr.
Rohini K. Srihari
William M. Stanton
Jennifer M. Suchin
Lynn M. Tranchell
Jason C. Van Blargan
Nicholas F. Vitulli
Diana K. Webster
Janyce M. Wiebe
Zhaomo Yang
Albert Hanyong Yuhan
Martin J. Zaidel

Stuart C. Shapiro

Contents

List of Figures

Chapter 1

Syntax

1.1 Notation

The syntax is given in this chapter using Extended Backus-Naur Form (EBF). Terminal symbols are surrounded by the quotation marks "`" and "'". Sequences of items are separated by commas, ",". Parentheses "(" and ")" are used as grouping brackets. Alternatives are separated by "|". Optional symbols are surrounded by "[" and "]". Material that can be repeated zero or more times is followed by "*". Material that can be repeated one or more times is followed by "+". Each syntactic rule is terminated by ";". Material starting with "//" and extending to the end of the line is a comment. The symbol $\not{b}$ appearing instead of a comma indicates that the two surrounding items are to appear without whitespace separating them; otherwise consecutive items must be identifiable to the reader as separate tokens. Items in *italics* are expected to be understood without definition herein. The characters i, j, and k stand for any non-negative integers such that $i \leq j \leq k$. Material in red has not yet been implemented.

1.2 Syntax of Well-Formed Terms

The language in which SNePS 3 well-formed terms are expressed is a version of Common Logic Interchange Format (CLIF) (**?**).

```
wft                 =  atomicwft
                       | 'wft' b̸ i
                       | '(' , function , argument+ , ')'
                       | '(' , binaryop , argument , argument , ')'
                       | '(' , naryop , wft* , ')'
                       | '(' , param2op , '(' , i , j , ')' wft+ , ')'
                       | '(' , 'thresh' , '(' , i , ')' wft+ , ')'
                       | '(', 'close' , (atomicname | ('(', atomicname+ , ')')) ,
                           wft , ')'
                       | '(', 'every' , atomicname , wft* , ')'
                       | '(', 'some' , atomicname , '(' , atomicname , ')' , wft* , ')'
                       | '(', '?' b̸ atomicname , wft* , ')'
                       | Generalized quantifiers to replace nexists ;

binaryop            =  'if' | i b̸ '=>' | 'v=>' ;

naryop              =  'and' | 'or' | 'not' | 'nor' | 'thnot' | 'thnor' | 'nand'
                       | 'xor' | 'iff' ;

param2op            =  'andor' | 'thresh' ;

atomicwft           =  atomicname | Lisp string | Lisp number ;

atomicname          =  Clojure symbol other than wfti ;

function            =  wft // other than reservedWord ;

argument            =  wft | 'nil' | '(' , argumentFunction , wft* , ')' ;

argumentFunction    =  'setof' ;

reservedWord        =  'every' | 'some' | 'close' | '?' b̸ atomicname
                       | binaryop | naryop | param2op ;
```

Every non-atomic `wft` (that is, a `wft` other than an `atomicwft`) is given a wft-name when it is stored into the SNePS knowledge base. The wft-name of every stored term may be seen by evaluating the user command `(list-terms)`. The user expression `wft`*i* is a syntactic abbreviation of the `wft` that was assigned `wft`*i* as its wft-name. If no `wft` has yet been assigned that wft-name, `wft`*i* is syntactically illegal.

1.3 Syntax of Paths

In this section is presented the syntax of path expressions used in `definePath` and `defineSlot`.

```
path  =  slotname
         | slotname b̸ '-'
         | '!'
         | '(' , 'converse' , path , ')'
         | '(' , 'kplus' , path , ')'
         | '(' , 'kstar' , path , ')'
         | '(' , 'compose' , path* , ')'
         | '(' , 'or' , path* , ')'
         | '(' , 'and' , path* , ')'
         | '(' , 'irreflexive-restrict' , path , ')'
         | '(' , 'domain-restrict' , '(' , path , wft , ')' , path , ')'
         | '(' , 'range-restrict' , path , '(' , path , wft , ')' , ')' ;
```

Chapter 2

Running CSNePS

CSNePS is implemented in the Clojure programming language, a recently developed dialect of lisp witch runs (primarily) on the Java Virtual Machine.

2.1 Pre-Requisites

- **git** is a versioning system which will be required. Install it via your preferred package manager if you do not already have it.
- **Leiningen** (aka **lein**) is a dependency manager for Clojure projects. To install Leiningen, follow the directions at `https://github.com/technomancy/leiningen`.

2.2 Running CSNePS at the Command Line

The preferred method of running CSNePS at the command line is to use the REPL component of Leiningen.

1. At the command prompt, change directories to the CSNePS directory. This is the directory containing the project.clj file which Leiningen will use to load the project.
2. Type `lein repl` at the command prompt and hit enter. Any required dependencies will be downloaded, Clojure will load, and you will be presented a Clojure prompt, `csneps.core.snuser=>`, indicating that CSNePS has loaded and we are in the 'csneps.core.snuser' namespace.

2.3 Running the CSNePS GUI

The CSNePS GUI may be started with or without pre-loading CSNePS. If you choose to pre-load CSNePS you will also have access to the command prompt in your terminal window. This will not be available of you choose not to pre-load CNSePS.

2.3.1 By Pre-Loading CSNePS

1. Follow the directions from Section 2.2.
2. Type `(startGUI)` from the prompt and press enter. This will load the CSNePS GUI.

2.3.2 Without Pre-Loading CSNePS

1. At the command prompt, change directories to the CSNePS directory. This is the directory containing the project.clj file which Leiningen will use to load the project.

2. Type `lein run` at the command prompt and hit enter. This will load CSNePS and its GUI.

2.4 Using Eclipse to Run CSNePS

Before getting started with Eclipse, be sure to check out the Sneps3-Clojure project, change directories to that project, and run `lein deps` to be sure the project and its dependencies are up to date.

In order to run and develop CSNePS in Eclipse, you will require a version of Eclipse which includes Java support. The version on the CSE servers seems somehow incompatable with CounterClockwise, so I recommend installing the latest version from `http://www.eclipse.org` in your home directory.

First we must install the CounterClockwise plugin, which lets us run Clojure apps from Eclipse.

1. Click the Help menu, then ”Install New Software...”

2. In the “Work With:” field, enter `http://ccw.cgrand.net/updatesite/` and click the “Add...” button.

3. Enter ”CounterClockwise” (or whatever text you like) in the “Name” field and click “OK”.

4. Check the box next to ”Clojure Programming” in the list that appears, and click the “Next” button.

5. In the dialogs that follow click Next, agree to the license agreement, and click Finish.

6. Agree to any warnings about unsigned content, and agree to restart Eclipse when installation has finished.

Now we will import our project.

1. Choose ”Import...” from the File menu.

2. Cloose ”Existing Project or Workspace” under the ”General” node in the tree shown.

3. With the ”Select root directory” radio button selected, click browse and browse to the SNePS3-Clojure directory.

4. Click next and follow the dialogs until it is imported.

Now the project is imported into Eclipse and we can run it. In the project explorer on the left side of the workspace, expand the SNePS3-Clojure node and then the src node, right click on core.clj, and choose to run it as a Clojure Application. Once the repl loads, run `(loadsneps3)`.

2.5 Using Emacs to Run CSNePS

2.5.1 Configuring Emacs

There are multiple methods for configuring Emacs to run and edit Clojure code. The method described here is the most desirable as of the time of this writing. These instructions require emacs 24, as it has a package facility we will use.

First, add the following to your emacs initialization file (by default `/.emacs.d/init.el`).

```
(require 'package)
(add-to-list 'package-archives
  '("marmalade" . "http://marmalade-repo.org/packages/"))
(add-to-list 'package-archives
  '("melpa-stable" . "http://melpa-stable.milkbox.net/packages/"))

(package-initialize)

(defvar my-packages '(clojure-mode
                      clojure-test-mode
                      cider))
```

Evaluate the buffer with the initialization file in order to download the required packages.

2.5.2 Loading and Running the Project in Emacs

1. Load Emacs and switch the current directory to the CSNePS directory. This is the directory containing the project.clj file which will be used to load the project.
2. Type `M-x cider-jack-in`. A Clojure REPL will be started in a new buffer, and you will be presented with a Clojure prompt, `csneps.core.snuser=>`, indicating we have started in the 'csneps.core.snuser' namespace.
3. CSNePS is now loaded and is ready to be used.

Chapter 3

User Commands

`(allTerms )` [*Function*]
Returns a set of all the terms in the knowledge base.

`(ask` *`expr`*`)` [*Function*]
Attempts to derive the term `expr` or its negation. Returns the derived term, or nil if it is not derivable in the current context. If the term is not derivable, focused inference is left running until it is derived, or the task is canceled using *cancel-infer-of* or *cancel-focused-infer*.

`(askif` *`expr`*`)` [*Function*]
Attempts to derive the term `expr`. Returns the derived term, or nil if it is not derivable in the current context. If the term is not derivable, focused inference is left running until it is derived, or the task is canceled using *cancel-infer-of* or *cancel-focused-infer*.

`(askifnot` *`expr`*`)` [*Function*]
Attempts to derive the negation of the term `expr`. Returns the derived term, or nil if it is not derivable in the current context. If the term is not derivable, focused inference is left running until it is derived, or the task is canceled using *cancel-infer-of* or *cancel-focused-infer*.

`(askwh` *`exprpat`*`)` [*Function*]
Returns a set of substitutions for variable placeholders for the term pattern *`exprpat`* that are currently derivable in the current context; or the empty set if there are none. If no instances are derivable, focused inference is left running until an instance is derivable, or the task is canceled using *cancel-infer-of* or *cancel-focused-infer*.

`(assert` *`expr`*`)` [*Function*]
Asserts the term expressed by *`expr`* in the current context.

`(assert!` *`expr`*`)` [*Function*]
Asserts the term expressed by *`expr`* in the current context, and triggers forward inference.

`(cancel-focused-infer )` [*Function*]
Cancels all focused inference tasks.

`(cancel-infer-from` *`exprpat`*`)` [*Function*]

Cancels any forward focused reasoning tasks deriving from *exprpat*.

`(cancel-infer-of exprpat)` [*Function*]
Cancels any focused reasoning tasks attempting to derive *exprpat*.

`(clearkb &optional (clearall nil))` [*Function*]
Reinitializes the SNePS knowledge base. If `clearall` is non-`nil` also reinitializes all slots, caseframes, and semantic types.

`(currentContext )` [*Function*]
Returns the current context.

`(defineCaseframe type frame &key docstring fsymbols)` [*Function*]
Defines a caseframe, where: `type` is the name of a SNePS semantic type; `frame` is either `(slot1 ... slotn)` or `('function-symbol slot1 ... slotn)`; `docstring` is a caseframe documentation string; `fsymbols` is a list of function symbols required if `first` of the `frame` is not quoted.

`(defineContext name &key (docstring "") (parents '(BaseCT)) hyps)` [*Function*]
Defines a new context with the given name, docstring, parent contexts, and initial hypotheses. If `dosctring` is omitted, it defaults to the empty string. If `parents` is omitted, it defaults to `'(BaseCT)`. If `hyps` is omitted, it defaults to the empty list.

`(definePath slotname path)` [*Function*]
Given a slot name, `slotname`, and a path expression, `path` (*see* §1.3), generate the functions that will compute that path and its converse, and store them in the slot named `slotname`.

`(defineSlot name &key type docstring posadjust negadjust min max path)` [*Macro*]

Defines the slot named `name`. `type` must be a semantic type. It defaults to `Entity`. `docstring` must be a string. It defaults to the empty string. `posadjust` must be either `reduce` (default), `expand`, or `none`. `negadjust` must be either `reduce`, `expand` (default), or `none`. `min` must be a positive integer. It defaults to `1`. `max` must be either `nil` (default) or an integer equal to or greater than `min`. `path` must be either `nil` (default) or a path (*see* §1.3).

`(defineTerm term &optional (semtype 'Entity))` [*Function*]
If `term` is not already a term in the SNePS knowledge base, it is added to the KB with the semantic type `semtype`, which defaults to `Entity`. If `term` is already a term in the KB with semantic type `currenttype`:

- if `currenttype` is a subtype of `semtype`, the type of `term` is left as is;
- if `semtype` is a subtype of `currenttype`, the semantic type of `term` is lowered to `semtype`;
- if `currenttype` and `semtype` have one greatest common subtype, the semantic type of `term` is changed to that type;
- if `currenttype` and `semtype` have several greatest common subtypes, the user is asked which one (s)he wants `term` to be, and `term`'s semantic type is changed to that type;
- otherwise, an error is generated.

The term is returned.

`(defineType newtype supers &optional docstring)` [*Macro*]
Defines *newtype* to be a SNePS semantic type, and a subtype of the types listed in the list *supers*. If *docstring* is given, it is set as the documentation string of the new type. Returns a string-message, either of success or what the problem was.

`(demo &key file pause)` [*Function*]
Echoes and evaluates the forms in the *file*. If *pause* is non-`nil` (the default is `nil`), will pause after echoing each form, but before evaluating it. If the *file* is omitted, a menu will be presented of available demos.

`(find exprpat)` [*Function*]
Returns a list of vector pairs. In each pair, the first element is an instance of *exprpat* in the knowledge base, and the second is a substitution, which when applied to *exprpat* would give that instance. *exprpat* may be any wft with variables, symbols starting with a "?", in the place of any subterms.

`(findTerm name)` [*Function*]
Returns the term named *name*, or `nil` if there isn't one. The name of an atomic term is a symbol, string, or number. The name of a molecular term is its wftname.

`(krnovice boolean)` [*Function*]
If set to `true` (the default value is `false`), slots and caseframes will automatically be created whenever a function symbol is used that is not already associated with a caseframe. The slots will be named `fn`, `arg1`, `arg2`, *etc.*, and both slots and caseframes will have their default parameters. This should only be used by novices, or for very quick tests, as the careful modeling required by defining types, slots, and caseframes might be ignored.

`(list-caseframes )` [*Function*]
Prints all the caseframes.

`(listContexts )` [*Function*]
Prints a list of all the contexts.

`(list-slots )` [*Function*]
Prints a list of all the SNePS slots.

`(list-terms &key (asserted nil) (types nil))` [*Function*]
Prints a list of all the terms in the KB. If `asserted` is non-null, only asserted propositions will be printed; otherwise, all terms will be printed. If `:types` is non-null, the type of each term will also be printed.

`(listkb )` [*Function*]
Prints the current context and all propositions asserted in it.

`(pathsfrom terms path)` [*Function*]
Returns the set of terms at the end of the given *path* (*see* §1.3) from *terms*, which must be a term, the name of a term, a list of terms or names of terms, or a set of terms.

`*PRECISION*` [*Variable*]
A positive integer: a floating point number will be rounded to this number of decimal places before being converted to a term.

`(remove-from-context` *`term ctx`*`)` [*Function*]
Removes the provided *term* from the context *ctx*. The term will still be asserted in contexts it isn't removed from.

`(sameFrame` *`newf oldf`*`)` [*Function*]
Associates the same frame associated with the function symbol *`oldf`* with the symbol, or list of symbols, *`newf`*.

`(setCurrentContext` *`ctx`*`)` [*Function*]
If *`ctx`* is a context name, makes the context named *`ctx`* the current context. If *`ctx`* is a context, makes it the current context. Else raises an error.

`(showTypes )` [*Function*]
Graphically displays all the defined semantic types.

`(startGUI &rest` *`terms`*`)` [*Macro*]
Starts the CSNePS GUI. Takes a variable number of *`terms`* to display on the graph. Each term is either found or defined using `defineTerm`. If no terms are given, the entire graph will be displayed.

`(unassert` *`prop`* `&optional (`*`cntxt`* `(currentContext)))` [*Function*]
Unasserts the proposition *`prop`* in the given context and all ancestor contexts. Currently there is no belief revision, so propositions derived using *`prop`* might still be asserted, and prop, itself, might be rederivable.

`(writeKBToTextFile` *`file`* `&optional` *`headerfile`*`)` [*Function*]
Writes the KB to the given text *`file`*, so that when that file is loaded, all the propositions asserted in the current KB will be asserted in the new KB. If the *`headerfile`* is included, a load of that file will be written before any of the asserts.

University at Buffalo Public License ("UBPL") Version 1.0

1. Definitions.

1.0.1. "Commercial Use"
means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor"
means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version"
means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code"
means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism"
means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable"
means Covered Code in any form other than Source Code.

1.6. "Initial Developer"
means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work"
means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License"
means this document.

1.8.1. "Licensable"
means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications"
means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

a. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

b. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code"
means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims"
means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code"
means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your")
means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

a. under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

b. under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

c. the licenses granted in this Section 2.1 (a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

d. Notwithstanding Section 2.1 (b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

a. under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

b. under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

c. the licenses granted in Sections 2.2 (a) and 2.2 (b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

d. Notwithstanding Section 2.2 (b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims

If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs

If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section 3.4 (a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.

You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Sections 3.1, 3.2, 3.3, 3.4 and 3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions

University at Buffalo ("UB") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by UB. No one other than UB has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works

If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "University at Buffalo", "University at BuffaloPL", "UBPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the University at Buffalo Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

6.4. Origin of License

This License is derived from the familiar Mozilla Public License Version 1.1 (“MPL”) and differs only in that 1) the title now refers to UB to indicate that UB is the licensor; 2) UB retains the sole right to publish revised versions of this license (See 6.1 and 6.2); 3) Section 6.3 now refers to the phrases “University at Buffalo”, “University at BuffaloPL”, “UBPL”; 4) the License shall be governed by law provisions of the state of New York and any litigation relating to this License shall be subject to the jurisdiction of the state and federal courts of the State of New York and all parties consent to the exclusive personal jurisdiction of those courts (See 11); and 5) Research Foundation of State University of New York, on behalf of University at Buffalo is cited as the copyright owner of the original code (See Exhibit A).

7. DISCLAIMER OF WARRANTY

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN “AS IS” BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. Termination

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as “Participant”) alleging that:

(a) such Participant’s Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant’s Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. government end users

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. Miscellaneous

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by law provisions of the state of New York (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the state and federal courts of the State of New York and all parties consent to the exclusive personal jurisdiction of those courts, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. Responsibility for claims

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. Multiple-licensed code

Initial Developer may designate portions of the Covered Code as “Multiple-Licensed”. “Multiple-Licensed” means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the UBPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

Exhibit A - University at Buffalo Public License.