The Konvax Hesperides development environment

Konvax Corp - Hesperides s.r.l.

Version 4.0., February the 12^{th}, 2018

Chapter 1

Introduction

This manual describes, using examples when necessary, the Konvax Hesperides development environment (©2010- Konvax and 2016- Hesperides), its architecture, its languages and its extention modules. Note that the *Konvax Hesperides development environment*, in the following *SDK* and it main tool *VAPP-Virtual Application* that gives the access to all the *SDK* components, is distributed, for portability reasons, as a docker container. For this reason we suggest to not change the localhost IP addresses written in the communication language module. The localhost IP address are internal to the container so they do not consume the ports and the IP of the host environment.

Chapter 2

The main modules

2.1 Communication Manager

This module is in charge to act as a server for HTTP, HTTS and TCP communications. It is completely configurable and runs in sinergy with the Access Manager. When the *VAPP-Virtual Application* is ran the first module that is executed is the Communication Manager and the one and only parameter that need to be passed on the *VAPP-Virtual Application* command line is the name of the file containing the configuration of the Communication Manager.
A typical run of the *VAPP-Virtual Application* is /path/*VAPP-Virtual Application* <configuration file>.

2.1.1 The configuration language

This language is a very simple one, structured in lines of text, splitted in $key = value; value; value; \cdots$. The keys and the suggested values are show in the following. Some values are left only for historical reasons and are not more used:

- LbUser_VA_VRules=*here the file containing a program written both in Access Rule Language and in Validation Rule Language (see below)*
- LbUser_VA_VReports=*the folder where all the Validation Reports generated by the Access Rule Language and in Validation Rule Language will be archived*
- LbUser_OutIp=0.0.0.0 *A sequence of IP numbers that will be randomly chosen by the Communication Manager to pass, under the control of one of the other modules, the communications to another server/peer. Application example: to add control access manager to a pre-existing Web Site. When the request has been correctly verified by the Access Manager, the request is passed to the destination Web Server. It is conceptually similar to a firewall or transparent/reverse proxy set as front-end to a DMZ*
- LbUser_AsDaemon=false *To daemonize the* VAPP-Virtual Application
- LbUser_Id_WhoAmI=VA *Do not change*
- LbUser_Initialize=127.0.0.1:61100 *Used internally by the* VAPP-Virtual Application *to connect to itself*
- LbUser_SVA_Listen= *Do not change*
- LbUser_FromVACommands= *Do not change*
- LbUser_Hy_VA_Sisters= *Do not change*
- LbUser_Hy_KAM_Children= *Do not change*
- LbUser_SecureVa= *Do not change*
- LbUser_VAObservedApps= *Do not change*
- LbUser_VA_GetAllHTTP= *Do not change*
- LbUser_VA_QueryFilters= *Do not change*
- LbUser_VA_HwAgentPort= *Do not change*
- LbUser_VA_HwAgentIp= *Do not change*

- LbUser_VA_SwAgentPort= *Do not change*
- LbUser_VA_SwAgentIp= *Do not change*
- LbUser_VA_AskForUrl= *Do not change*
- LbUser_VA_LocalSH_on=false *When true the internal Observation Manager is run sending the Observations to the Validation Manager.*
- LbUser_VA_TraceDomain=127.0.0.1:61107 *Do not change*
- LbUser_VA_KAM_CommandPort=61108 *Do not change*
- LbUser_VA_KAM_CommandIp=127.0.0.1 *Do not change*
- LbUser_ListenIp=127.0.0.1 *The main IP the* VAPP-Virtual Application *is listening on for HTTP protocol*
- LbUser_ListenPort=9999 *The main PORT the* VAPP-Virtual Application *is listening on for HTTP protocol*
- LbUser_VListenIpPort= #10.211.55.1:1935;192.168.1.253:8888 *Usually empty. If a list of semicolons separated additionals IP:PORT is given, the* VAPP-Virtual Application *will listen on these IPs:PORTs for HTTP connection. The* VAPP-Virtual Application *languages can read the IP:PORT of any communication*
- LbUser_VTCPPort #10.211.55.1:1935;192.168.1.253:8888 *Usually empty. If a list of semicolons separated IP:PORT is given, the* VAPP-Virtual Application *will listen on these IPs:PORTs for any TCP connection (ie: binary protocols, special protocol to be implemented inside one f the* VAPP-Virtual Application *languages, ...)*
- LbUser_VSSLPort=#0.0.0.0:4443:certs/certs.pem:certs/private.pem—TLS—HIGH,MEDIUM,!aNULL,!MD5:www.exa *A list of semicolons separated strings, like the one shown here, that specify all the parameters to start listening for HTTPS/SSL connections: listen ip, listen port, X509 cert, private key list of secure protocols accepted (like the configuration string of Apache web server), the server domain name (TLS SNI)*
- LbUser_HowManyInstances= *The* VAPP-Virtual Application *is event based, so with only a task can manage thousand and thousad communications in parallel. Anyway, on a multicore machine you can specify how many instances (threads) you want to launch. Consider that for any pair IP:PORT you specify in LbUser_VListenIpPort, LbUser_VTCPPort, LbUser_VSSLPort, a different thread is launched. So, if you specify N pairs IP:PORT and specify M instances in LbUser_HowManyInstances, $N*M$ threads will be launched.*
- LbUser_ThPoolSize=4 *Every thread of the* VAPP-Virtual Application *manages the communications using an event bases approach. The poolsize is a way to accept that inside every thread you can have a queue of N, in this case* 4*, communications waiting to be executed. This could be useful when some tasks programmed in the* VAPP-Virtual Application *could be cpu consuming.*
- LbUser_LspCode=test.scm *The file name of the GUILE (Lisp SCHEME) program linked to the* VAPP-Virtual Application*. SCHEME is one of the languages on which the* VAPP-Virtual Application *is based. It is a special instance of the Scheme (GUILE) linked to the* VAPP-Virtual Application *and extended to operate integrated to the* VAPP-Virtual Application
- LbUser_LogFile= *Do not change*
- LbUser_NewCall=true *Do not change*
- LbUser_Rules= *Do not change*
- LbUser_BlockMessage= *Do not change*

2.2 Observation Manager

The observation manager is a local (inside the *VAPP-Virtual Application*) module that, when enabled via the config file above described, analyzes all the commmunications in/out to the *VAPP-Virtual Application*, builds a short, structured data block, containing all the information related to the communication, and feeds the Validation Manager.

2.3 Access Manager

The Access Manager is activated for each communication that reaches the *VAPP-Virtual Application*. The running context is managed per TCP session level. Access Rules are executed to manage the communication.

2.4 Validation Manager

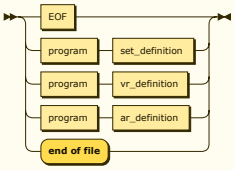
The Validation Manager is activated for each communication that reaches the *VAPP-Virtual Application*, coming from the Observation Manager. The running context is managed at global level (it is possible to operate for multiple TCP sessions). Validation Rules are executed for each observed communication and Validation Reports are issued (on file or on RDBMS) and the rules of the Access Manager can be enabled or disabled. In the following the *ARS* language, the language of the Access Manager (access rules) and of the Validation Manager (validation rules) is described and examples are given.

2.5 The ARS language of Access Manager and of Validation Manager

2.5.1 The syntax specification

Here you can find the syntax specification of the ARS language.

program:



referenced by:

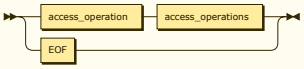
- program

ar_definition:

referenced by:

- program

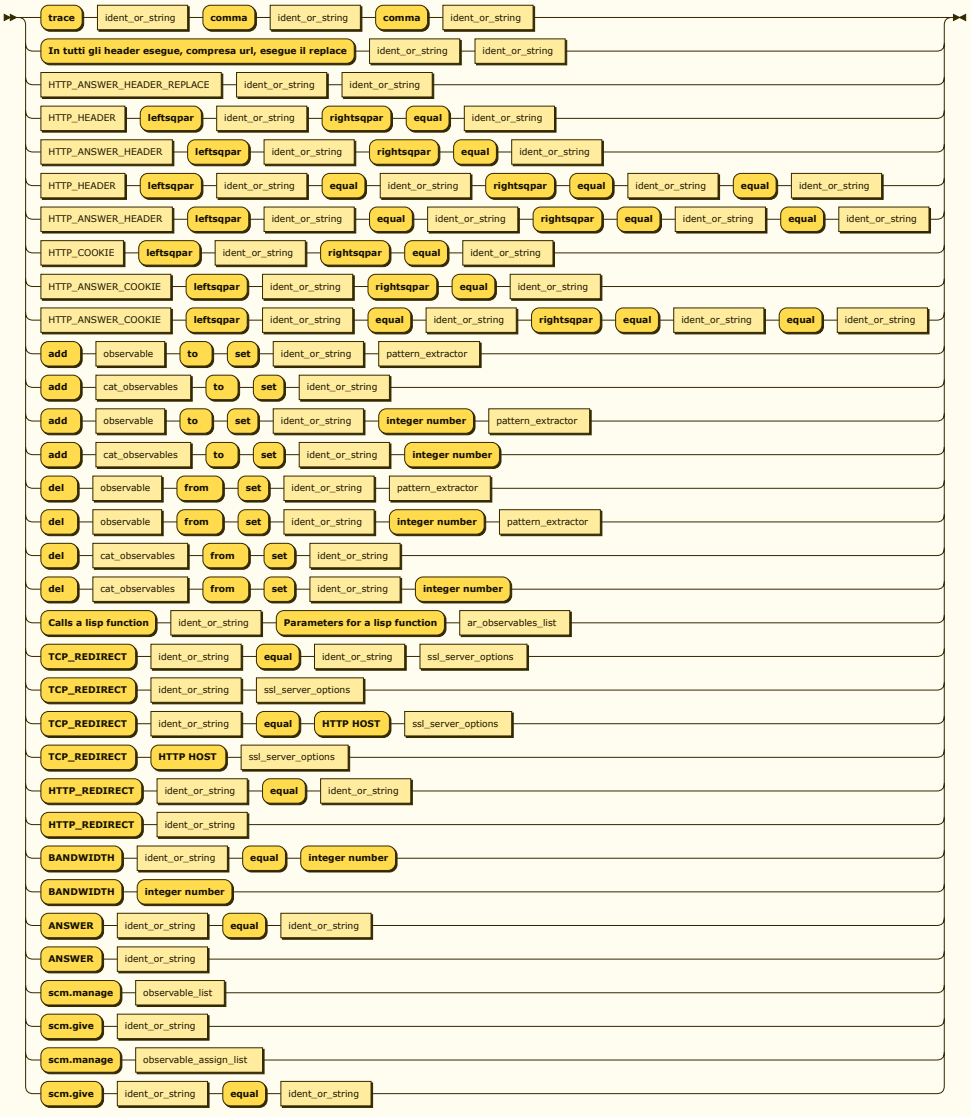
access_operations:



referenced by:

- access_operations
- ar_definition

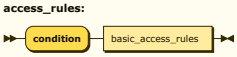
access_operation:



referenced by:

- access_operations

access_rules:



referenced by:

- ar_definition

vr_definition:

referenced by:

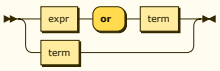
- program

clauses:

referenced by:

- clauses
- factor
- vr_definition

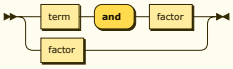
expr:



referenced by:

- clauses
- expr

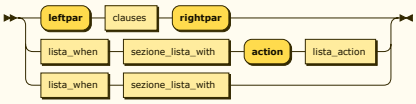
term:



referenced by:

- expr
- term

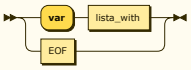
factor:



referenced by:

- term

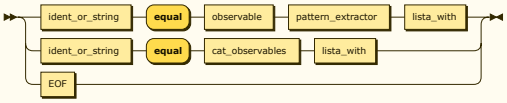
sezione_lista_with:



referenced by:

- factor

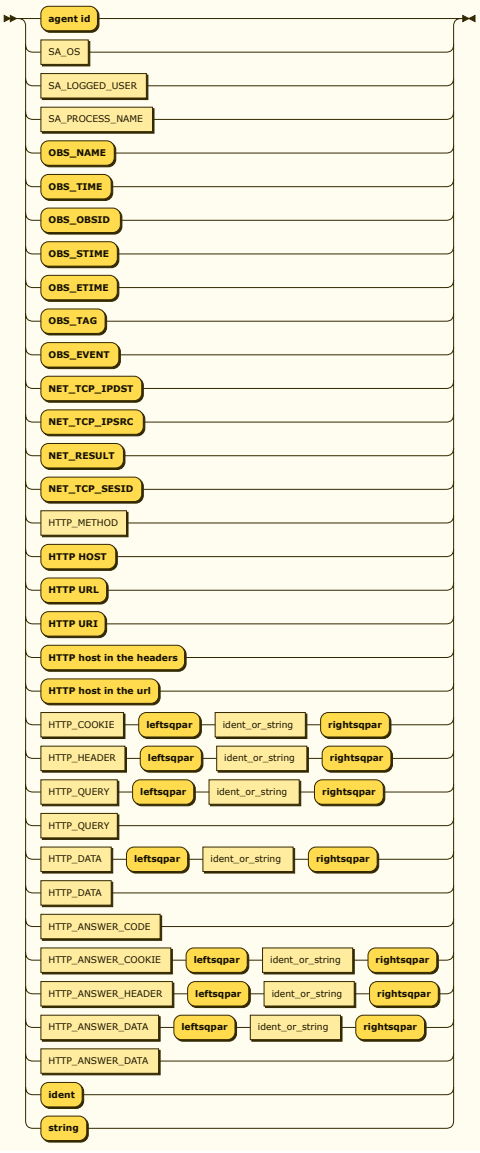
lista_with:



referenced by:

- lista_with
- sezione_lista_with

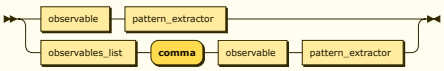
observable:



referenced by:

- access_operation
- item_action
- lista_with
- observable_assign_list
- observable_list
- observables_list

observables_list:



referenced by:

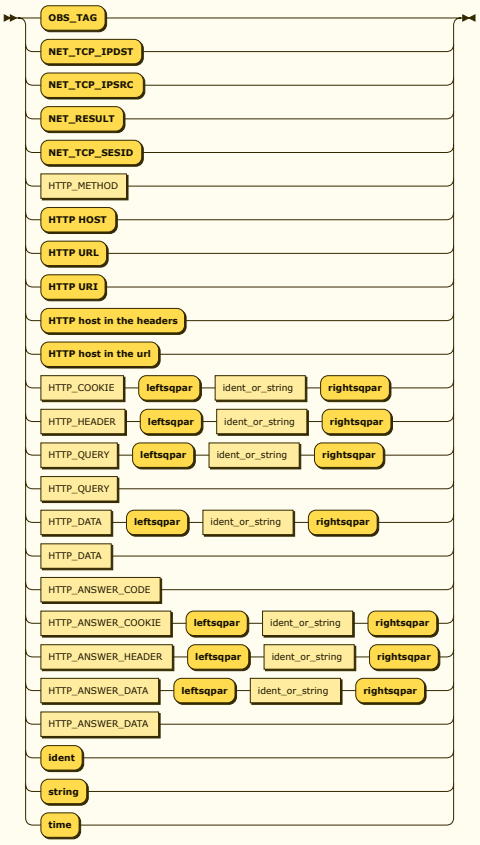
- basic_item_when
- cat_observables
- item_action
- observables_list

cat_observables:

referenced by:

- access_operation
- action_variable_item
- basic_item_when
- item_action
- lista_with

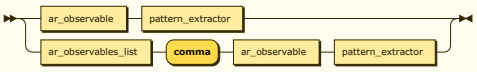
ar_observable:



referenced by:

- ar_observables_list

ar_observables_list:



referenced by:

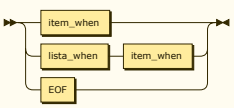
- access_operation
- ar_cat_observables
- ar_observables_list
- item_basic_access_rule_with_not

ar_cat_observables:

referenced by:

- item_basic_access_rule_with_not

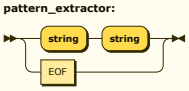
lista_when:



referenced by:

- factor
- lista_when

pattern_extractor:

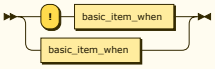


referenced by:

- access_operation
- action_variable_list
- ar_observables_list
- basic_item_when
- item_action

- item_basic_access_rule_with_not
- lista_action
- lista_with
- observables_list

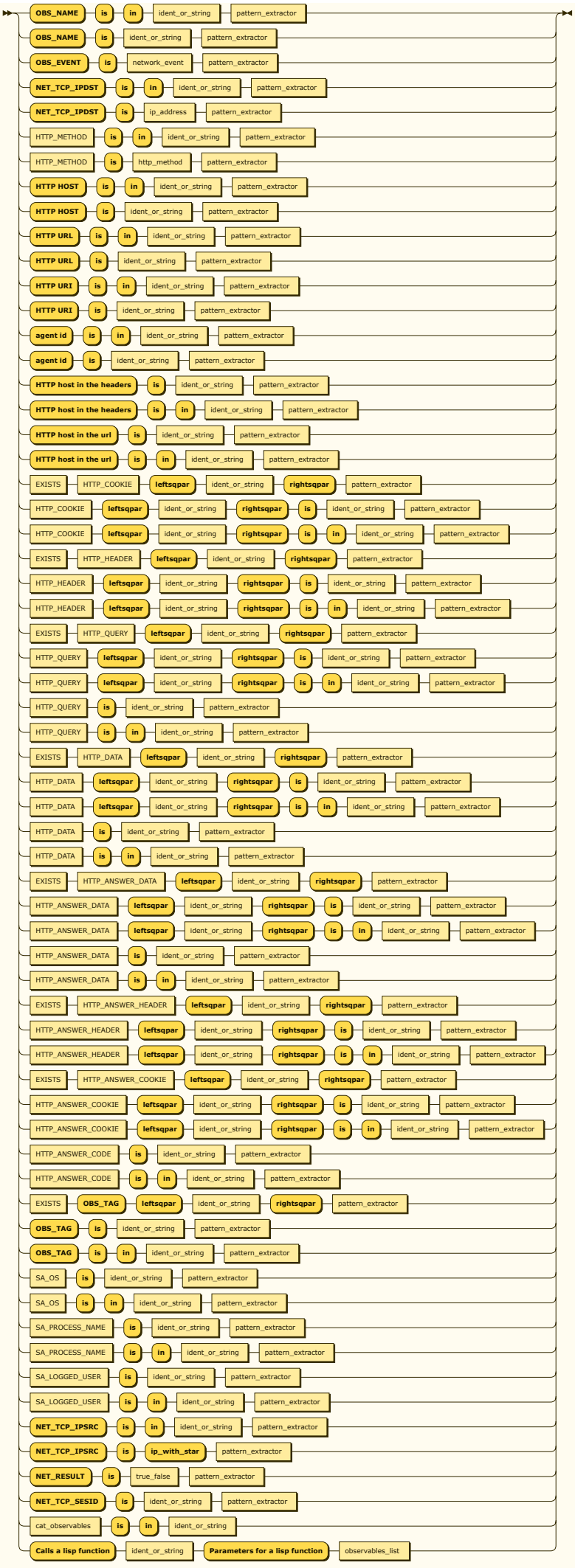
item_when:



referenced by:

- lista_when

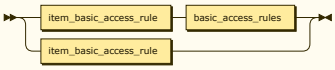
basic_item_when:



referenced by:

- item_when

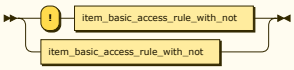
basic_access_rules:



referenced by:

- access_rules
- basic_access_rules

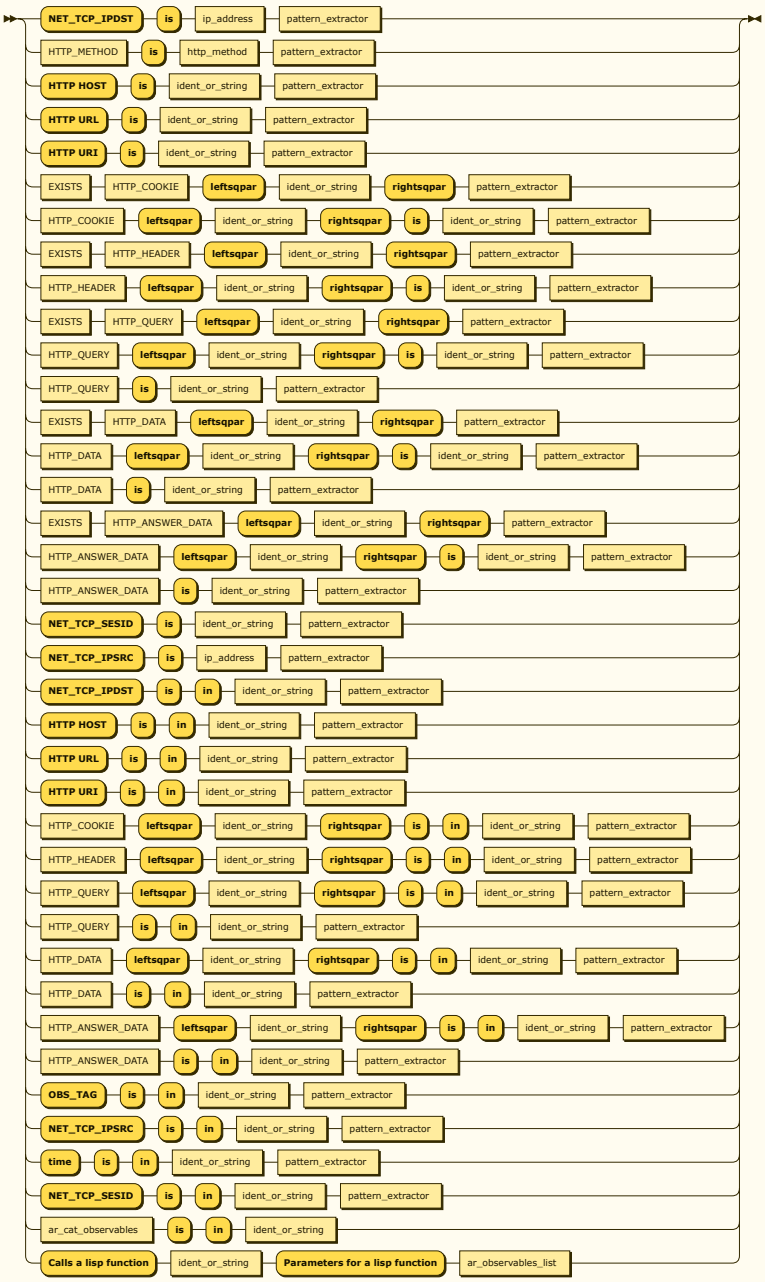
item_basic_access_rule:



referenced by:

- basic_access_rules

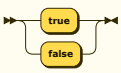
item_basic_access_rule_with_not:



referenced by:

- item_basic_access_rule

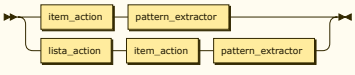
true_false:



referenced by:

- basic_item_when

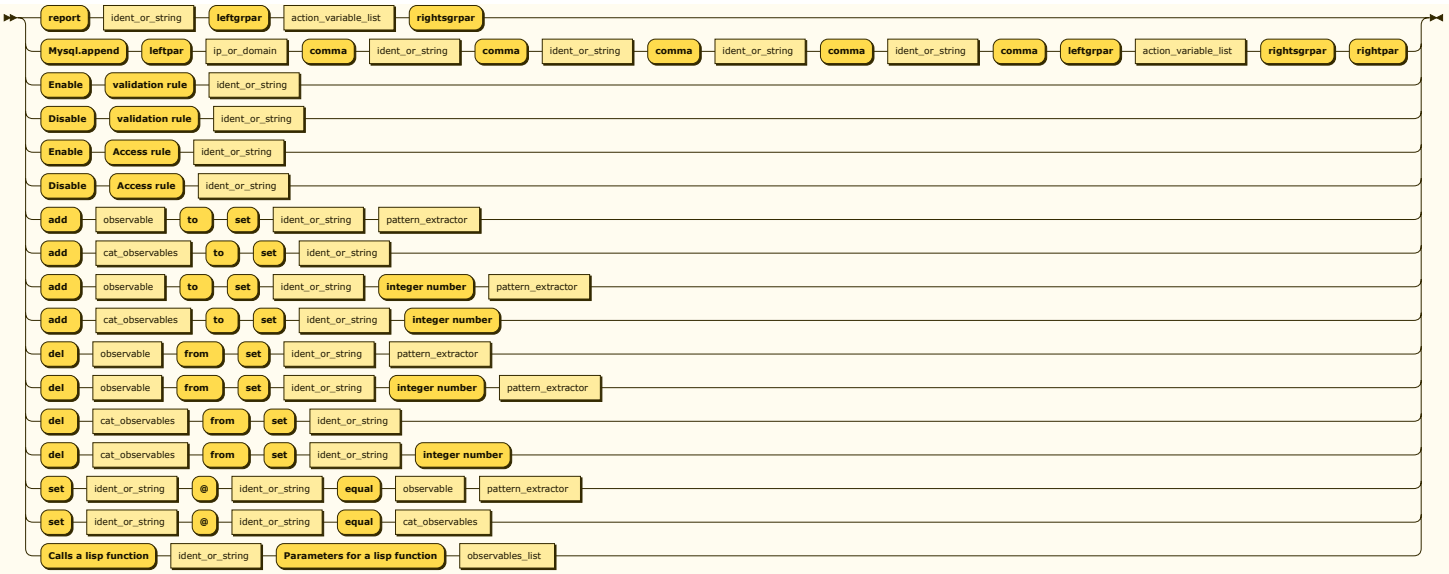
lista_action:



referenced by:

- factor
- lista_action

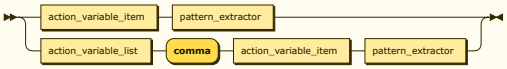
item_action:



referenced by:

- lista_action

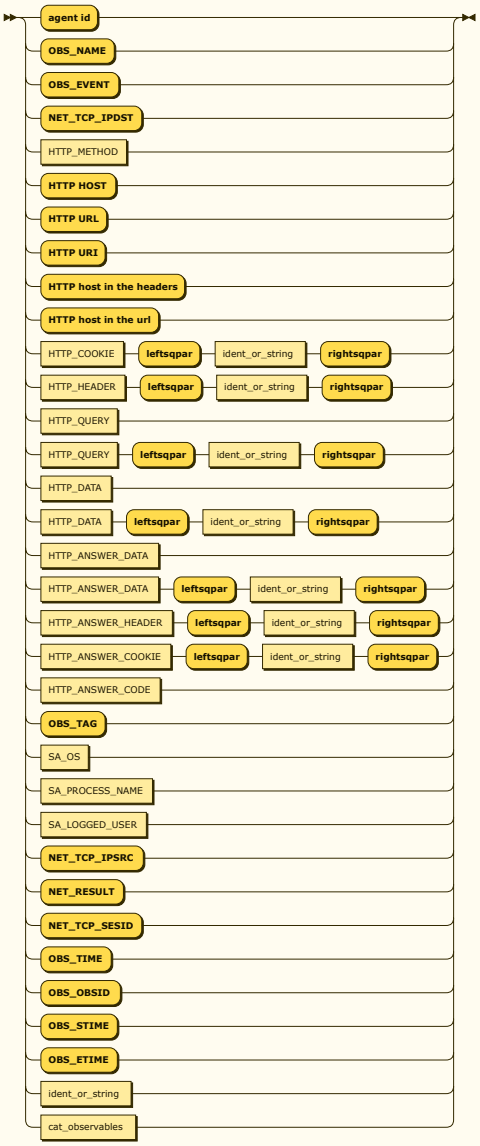
action_variable_list:



referenced by:

- action_variable_list
- item_action

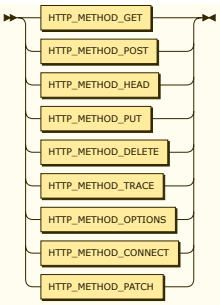
action_variable_item:



referenced by:

- action_variable_list

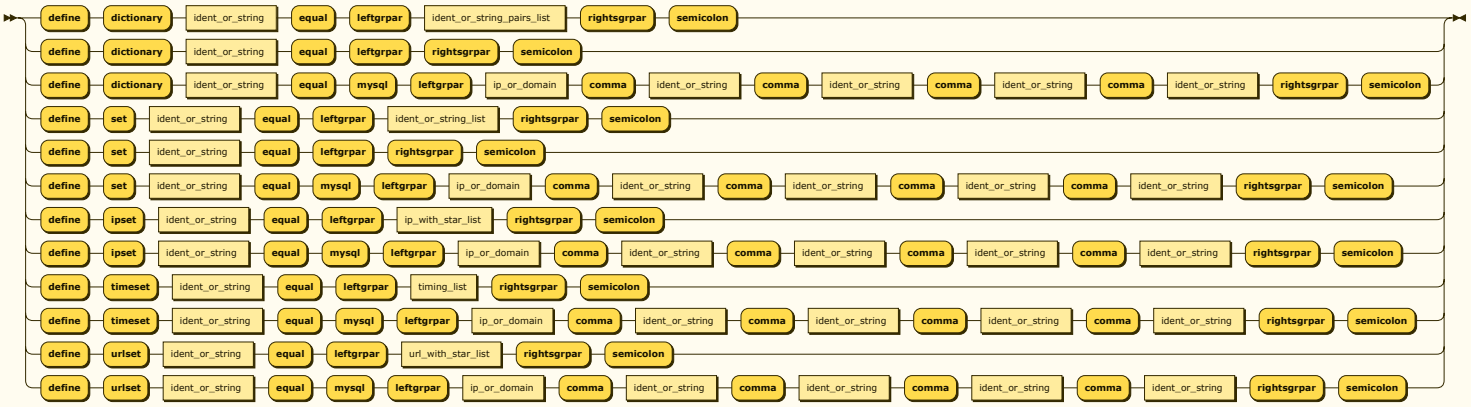
http_method:



referenced by:

- basic_item_when
- item_basic_access_rule_with_not

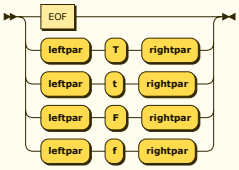
set_definition:



referenced by:

- program

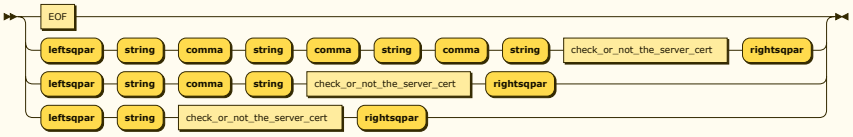
check_or_not_the_server_cert:



referenced by:

- ssl_server_options

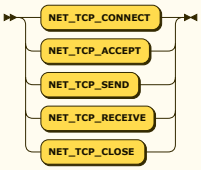
ssl_server_options:



referenced by:

- access_operation

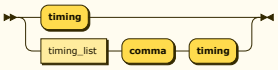
network_event:



referenced by:

- basic_item_when

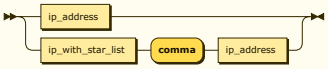
timing_list:



referenced by:

- set_definition
- timing_list

ip_with_star_list:



referenced by:

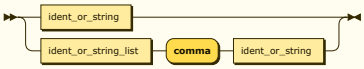
- ip_with_star_list
- set_definition

url_with_star_list:

referenced by:

- set_definition
- url_with_star_list

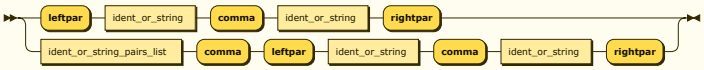
ident_or_string_list:



referenced by:

- ident_or_string_list
- set_definition

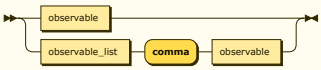
ident_or_string_pairs_list:



referenced by:

- ident_or_string_pairs_list
- set_definition

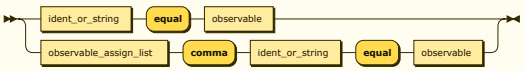
observable_list:



referenced by:

- access_operation
- observable_list

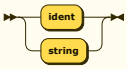
observable_assign_list:



referenced by:

- access_operation
- observable_assign_list

ident_or_string:



referenced by:

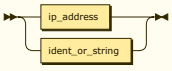
- access_operation
- action_variable_item
- ar_definition
- ar_observable
- basic_item_when
- ident_or_string_list
- ident_or_string_pairs_list
- ip_or_domain
- item_action
- item_basic_access_rule_with_not
- lista_with
- observable
- observable_assign_list
- set_definition
- vr_definition

ip_address:

referenced by:

- basic_item_when
- ip_or_domain
- ip_with_star_list
- item_basic_access_rule_with_not

ip_or_domain:



referenced by:

- item_action
- set_definition

The file *ars.xhtml* is the interactive version of the syntax specification.

2.5.2 The lexical specification

On the left the way the tokens are written in the language, on the right the tokens specified in the syntax specification.

- *string*: any sequence of characters delimited by a pair of "". it is exactly like the C language.
- *number*: any sequence of digits, decimal, octals, hexadecimals like the C language.
- *comment*: single line: //, like the C language
- *comment*: multiple lines (* ... *), like the C language except that instead of /* ... */ the (* ...*) are used.
- !: not
- *not*: not
- *or*: or
- *and*: and
- *set*: set
- *urlset*: urlset
- *ipset*: ipset
- *next*: next
- *var*: var
- *cat*: cat
- *call*: call
- *with*: with
- *enabled*: enabled
- *disabled*: disabled
- *default*: default
- *when*: when
- *tcp.redirect*: TCP_REDIRECT
- *http.redirect*: HTTP_REDIRECT
- *answer*: answer
- *define*: define
- *mysql*: mysql
- *true*: true
- *false*: false
- *vr*: vr
- *key*: key
- *action*: action
- *is*: is
- *condition*: condition
- *sequence*: sequence

- *of*: of
- *in*: in
- *always*: always
- *add*: add
- *del*: del
- *to*: to
- *from*: from
- *http.host*: HTTP_HOST
- *http.url.host*: HTTP_URL_HOST
- *http.header.host*: HTTP_HEADER_HOST
- *NET.SEND*: NET_TCP_SEND
- *NET.RECV*: NET_TCP_RECV
- *NET.ACCEPT*: NET_TCP_ACCEPT
- *NET.CONNECT*: NET_TCP_CONNECT
- *NET.CLOSE*: NET_TCP_CLOSE
- *NET.IPSRC*: NET_TCP_IPSRC
- *NET.IPDST*: NET_TCP_IPDST
- *NET.SESID*: NET_TCP_SESID
- *NET.RESULT*: NET_RESULT
- *obs.event*: OBS_EVENT
- *obs.name*: OBS_NAME
- *obs.time*: OBS_TIME
- *obs.obsid*: OBS_OBSID
- *obs.stime*: OBS_STIME
- *obs.etime*: OBS_ETIME
- *obs.tag*: OBS_TAG
- *http.url*: HTTP_URL
- *http.uri*: HTTP_URI
- *manage*: SCM_MANAGE
- *give*: SCM_GIVE
- *HTTP.METHOD*: HTTP_METHOD
- *HTTP.GET*: HTTP_METHOD_GET
- *HTTP.POST*: HTTP_METHOD_POST
- *HTTP.HEAD*: HTTP_METHOD_HEAD
- *HTTP.PUT*: HTTP_METHOD_PUT
- *HTTP.DELETE*: HTTP_METHOD_DELETE
- *HTTP.TRACE*: HTTP_METHOD_TRACE

- *HTTP.OPTIONS*: HTTP_METHOD_OPTIONS
- *HTTP.CONNECT*: HTTP_METHOD_CONNECT
- *HTTP.PATCH*: HTTP_METHOD_PATCH
- *HTTP.ANSWER.HEADER.REPLACE*: HTTP_ANSWER_HEADER_REPLACE
- *HTTP.HEADER.REPLACE*: HTTP_HEADER_REPLACE
- *HTTP.COOKIE*: HTTP_COOKIE
- *HTTP.HEADER*: HTTP_HEADER
- *HTTP.PORT*: HTTP_PORT
- *HTTP.QUERY*: HTTP_QUERY
- *HTTP.DATA*: HTTP_DATA
- *HTTP.ANSWER*: HTTP_ANSWER
- *HTTP.ANSWER.DATA*: HTTP_ANSWER_DATA
- *HTTP.ANSWER.CODE*: HTTP_ANSWER_CODE
- *HTTP.ANSWER.HEADER*: HTTP_ANSWER_HEADER
- *HTTP.ANSWER.COOKIE*: HTTP_ANSWER_COOKIE
- *SA.id*: AGENT_ID
- *SA.OS*: SA_OS
- *SA.USER*: SA_LOGGED_USER
- *SA.PROCESS*: SA_PROCESS_NAME
- *SA.PROCESS.ATTACH*: SA_PROCESS_ATTACH
- *REPORT*: REPORT
- *MREPORT*: MREPORT
- *EXISTS*: EXISTS
- *DATA*: DATA
- *MAKE*: MAKE
- *AR*: AR
- *mysql.append*: MYSQL_APPEND
- *mysql.update*: MYSQL_UPDATE
- *ENABLE*: ENABLE
- *DISABLE*: DISABLE
- *KV*: KV
- `[a-zA-Z_][a-zA-Z0-9_+.-]*`: IDENT
- `/([a-zA-Z0-9_.~-]*)|\*)+`: URL_WITH_STAR
- `[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}(:[0-9]+)?`: IP_WITHOUT_STAR
- `([0-9]{1,3}|\*)\.([0-9]{1,3}|\*)\.([0-9]{1,3}|\*)\.([0-9]{1,3}|\*)(:([0-9]+|\*))?`: IP_WITH_STAR
- `)`: RIGHTPAR
- `[`: LEFTSQPAR

-]: RIGHTSQPAR
- {: LEFTGRPAR
- }: RIGHTGRPAR
- ;: SEMICOLON
- ,: COMMA
- =: EQUAL
- >: GT
- <: LT
- >=: GTE
- <=: LTE
- +: PLUS
- -: MINUS
- .: DOT
- $: DOLLAR
- @: AT

2.6 Some examples of the ARS language

Note that the Access Rules are executed in the order they have been written. When one matches, no more access rules are examined (for the current communication). The Validation Rules, instead, are all checked for matching and executed if matched. Remember that the name of the file containing the ARS programs must be written in the *VAPP-Virtual Application* configuration file, associated to the key: LbUser_VA_VRules.

```
//some set of url
DEFINE urlset rootsite = { /, /* }; // */
DEFINE urlset api_rootsite = { /api, /api/* }; // */
DEFINE urlset api_json = { /json, /json/* }; // */
DEFINE urlset give_answer = { /ans, /ans/* }; // */

//some set of strings
DEFINE set users_ok = { "none" } ;
DEFINE set set_login = { "login" };
DEFINE set set_commands = { "add", "sub", "mul", "div", "sqrt" };

//some set of IPS
DEFINE ipset set_full_ips = { 10.211.55.1:80, 10.211.56.3:40, 10.211.56.3:8080 }
DEFINE ipset set_class_ips = { 10.211.57.1:*, 10.211.58.*:* }

//A simple Access rule that answers "Hi" to the
//client when the requested url is in the set give_answer
DEFINE AR "answer"
        CONDITION
                http.url is in give_answer
        ACTION
                ANSWER "Hi"
        ;

//A simple Access Rule that activates the function associated
//to JSON in the Scheme code integrated in the \emph{VAPP-Virtual Application} when the
//requested url is in the set api_json
DEFINE AR "json"
```

```
        CONDITION
                http.url is in api_json
        ACTION
                MANAGE "JSON"
        ;

//A simple Access Rule that redirects the communications
//to the IP 10.211.55.180:32000 when the requested url
//is in the set api_rootsite and one of the fields of the URI request
//is name "cmd" and the associated key is in the set set_login
DEFINE AR "services management"
        CONDITION
                http.url is in api_rootsite
                http.query["cmd"] is in set_login
        ACTION
                tcp.redirect "10.211.55.180:32000"
        ;

//Specifies an empty set of strings
define set set_try = { };

//Similar to the previous access rule but the "username" key/value
//set in the URI must be in the set users_ok
DEFINE AR "services management ok"
        CONDITION
                http.url is in api_rootsite
                http.query["cmd"] is in set_commands
                http.query["username"] is in users_ok
        ACTION
                tcp.redirect "10.211.55.180:32000"
        ;

//A simple Access Rule showing how it is possible to modify
//its code from a Validation Rule. in the following a validation
//rule will change the value of variable <v1> and the communications
//will be redirected to the ip:port written in the <v1> instead
//to go to ip 128.97.27.37:80
DEFINE AR "GOTOucla.edu"
        CONDITION
                http.url is in rootsite
        ACTION
                TCP.REDIRECT v1="128.97.27.37:80"
        ;


//This access rule, matching for uri in the set api_rootsite, can be
//used as "dig" where to redirect all the api requests not satisfying
//the previous access rules.
//A simple answer is sent to the client.
DEFINE AR "services management all"
        CONDITION
                http.url is in api_rootsite
        ACTION
                ANSWER "<!DOCTYPE html><html><head><title> The Answer </title>
                </head><body>ok?</body></html>"
        ;

//Now a simple validation rule. For every pair client=>\emph{VAPP-Virtual Application}=>client
//some information are appended to the file named <log>
DEFINE VR "PrintAll"
        CONDITION
```

```
                OBS.EVENT is NET.SEND
        VAR
                //in this way it is possible to associate the opposite communication
                v_sid = net.sesid
        ACTION
                REPORT log {HTTP.HOST, HTTP.URI}
        NEXT
                OBS.EVENT is NET.RECV
                net.sesid is v_sid
        ACTION
                REPORT log {"DATA: " , HTTP.ANSWER.HEADER["Content-Type"]}
        ;


//This Validation rule recognizes a simple "login" request
//and the answer. If the answer is correct the user is temporary
//added to the set  users_ok for 30 seconds.
//CAT is for catenate, so all the data inside CAT {}
// are catenated in a big string
DEFINE VR "add user"
        CONDITION
                obs.event is net.send
                http.url is in api_rootsite
                http.query["cmd"] is in set_login
        VAR
                username = http.query["p1"]
                v_sid = net.sesid
        ACTION
                REPORT log { "Login request", v_sid, http.query, http.data }
        NEXT
        (
                        obs.event is net.recv
                        http.answer.code is "200 OK"
                        net.sesid is v_sid
                        http.answer.data["result"] is "#t"
                ACTION
                        add CAT { username } to set set_try
                        add username to set users_ok 30
                        REPORT log { CAT {"0: ", username, " ", http.answer.code, " ",
                        http.answer.data["result"]} }
                OR
                        obs.event is net.recv
                        http.answer.code is "200 OK"
                        net.sesid is v_sid
                        http.answer.data["result"] is "#f"
                ACTION
                        REPORT log { CAT {"1: " , username, " ", http.answer.code, " ",
                        http.answer.data["result"]} }
        )
        ;

//Some VR for managing errors
DEFINE SET errors = {};
DEFINE VR "USER-OK"
        CONDITION
                obs.event is net.send
                http.url is in api_rootsite
                http.query["username"] is in users_ok
        ACTION
                DEL CAT{http.query["username"], "-1"} from set errors
                DEL CAT{http.query["username"], "-2"} from set errors
```

```
                DEL CAT{http.query["username"], "-3"} from set errors
                DEL CAT{http.query["username"], "-4"} from set errors
                REPORT log { CAT { "Dropped all previous errors" } }
                //This is where the behaviour of the AR "GOTOucla.edu"
                //is modified. When this VR is executed, the AR "GOTOucla.edu"
                //redirects to the ip 128.97.27.37:8080
                SET "GOTOucla.edu"@v1="128.97.27.37:8080"

                //Ainside a VR it is possible enable or disable any AR, by name
                //enable  AR "GOTOucla.edu"
                //disable AR "GOTOucla1.edu"
        ;

//Another complex code to manage errors in the communication protocol
DEFINE VR "USER-ERROR-FINAL"
        CONDITION
                obs.event is net.send
                http.url is in api_rootsite
                exists http.query["username"]
                !http.query["username"] is in users_ok
                CAT{http.query["username"], "-4"} is in errors
        ACTION
                REPORT log { CAT {
                        " Cybersecurity attack coming from ",
                        net.ipsrc, " => ",
                        http.query["username"]
                        }
                }
                //SET "GOTOucla.edu"@v1="40.79.78.1:80"
                //disable AR "GOTOucla.edu"
                //enable  AR "GOTOucla1.edu"
        OR
                obs.event is net.send
                http.url is in api_rootsite
                exists http.query["username"]
                !http.query["username"] is in users_ok
                CAT{http.query["username"], "-3"} is in errors
        ACTION
                DEL CAT{http.query["username"], "-3"} from set errors
                ADD CAT{http.query["username"], "-4"} to set errors
        OR
                obs.event is net.send
                http.url is in api_rootsite
                exists http.query["username"]
                !http.query["username"] is in users_ok
                CAT{http.query["username"], "-2"} is in errors
        ACTION
                DEL CAT{http.query["username"], "-2"} from set errors
                ADD CAT{http.query["username"], "-3"} to set errors
        OR
                obs.event is net.send
                http.url is in api_rootsite
                exists http.query["username"]
                !http.query["username"] is in users_ok
                CAT{http.query["username"], "-1"} is in errors
        ACTION
                DEL CAT{http.query["username"], "-1"} from set errors
                ADD CAT{http.query["username"], "-2"} to set errors
        OR
                obs.event is net.send
                http.url is in api_rootsite
```

```
        exists http.query["username"]
        !http.query["username"] is in users_ok
ACTION
        ADD CAT{http.query["username"], "-1"} to set errors
;
```

Chapter 3

The scheme guile extensions

All the operative modules of the *VAPP-Virtual Application* have been implemented as guile extensions. Data structures, neural networks, image manipulation, persistent structures, are only some of these modules. In the following all the modules and their functions are described. A deep knowledge of the Scheme/GUILE language is mandatory to read this chapter.

3.1 module utils

- mtfa-nop: not operation
- **Number theory functions**
- mtfa-mpz-probab-prime-p: (mtfa-mpz-probab-prime-p <number to test> <how many tries>) => #t/#f
- mtfa-mpz-nextprime: (mtfa-mpz-nextprime <number>) => the next prime after number
- mtfa-mpz-gcdext: (mtfa-mpz-gcdext <num1> <num1>) => the extended gcd between num1 and num2
- mtfa-mpz-invert: (mtfa-mpz-invert <num1> <module>) => the inverted module of num1 ih ehf dhf sudhf isudhf isuhdf isuhdf isuhdf isuhdf
- mtfa-mpz-factorize: (mtfa-mpz-factorize <num> <prove primality>) => the factorization of num. Stops when prove primality is exceeded
- mtfa-chinese-remainder : (mtfa-chinese-remainder <list of primes> <list of modules>) => the solution of the chinese reminder problem
- mtfa-http-get-page: (mtfa-http-get-page host port proto method url headers data proxy-name proxy-port timeout) => a cons mafe by the http headers and http body of the requested url
- fs-io-to-bv: (fs-io-to-bv <file name>) => returns file content as a bytevector
- fs-io-from-bv : (fs-io-from-bv <file name> <bytevector>) => writes in binary a bytevector to a file
- fs-io-to-string: (fs-io-to-string <file name>) => returns a string containing the complete file content
- fs-io-to-list: (fs-io-to-list <file name>) => returns a list of strings, one per file line
- fs-io-to-vector: (fs-io-to-vector <file name>) => like fs-io-to-list but returns a vector
- fs-io-from-string: (fs-io-from-string <fn> <str>) => write a string to a file
- crlf-to-lf: (crlf-to-lf <str>) => substitute crlf to lf in a string
- mtfa-howmany-files: (mtfa-howmany-files <folder>) => returns how many files are in a folder
- mtfa-rand-mat-range: (mtfa-rand-mat-range low high) => a strong random number betwwen low and high
- mtfa-rand-range: (mtfa-rand-range low high) => a weak random number betwwen low and high

- mtfa-permute: (mtfa-permute <list>) => permutes list items
- OpenDb: (OpenDb ip user passwd name port) => pointer to the mysql db opened
- CloseDb: (CloseDb <db>) => closes the mysqldb pointed by <db>
- mtfa-fs3-make: (mtfa-fs3-make) => return a very fast key/value data structure (ds). The key must be a string
- mtfa-fs3-add: (mtfa-fs3-add <ds> <key> <value>) => insert a key/value pair in the ds data structure
- mtfa-fs3-get: (mtfa-fs3-get ds key) => return the value associated to key
- mtfa-fs3-update: (mtfa-fs3-update ds key new-value) => add a new value to the key/value pair, replace the previous
- mtfa-fs3-serialize: (mtfa-fs3-serialize ds) => serializes in a bytevector the fs3 data structure
- mtfa-fs3-deserialize: (mtfa-fs3-deserialize <byte vector>) => deserialized a serialized fs3 in a new fs3 data structure
- mtfa-fs3?: (mtfa-fs3? ds) => is ds an fs3 data structure?
- mtfa-fs3-get-all: (mtfa-fs3-get-all ds) => returns all the k/v contained in the fs3
- mtfa-fs3-for-each: (mtfa-fs3-for-each ds <two par function>) => apply the function to all the (key value) in the fs3 data structure
- mtfa-ph-make: (mtfa-ph-make) => a perfect hash data structure.
- mtfa-ph-add: (mtfa-ph-add ds key) => add a key in the ds. returns a unique number
- mtfa-ph-get: (mtfa-ph-get ds key) => returns the unique number associated to the key
- mtfa-ph-get-val: (mtfa-ph-get-val ds index) => returns the key whose unique id is index
- mtfa-ph-get-all: (mtfa-ph-get-all ds) => returns a list of pairs <key . index>
- mtfa-ph-for-each: (mtfa-ph-for-each ds <two pars function>) => the function is called with pars <index key> for all the items in the ds
- mtfa-ph-serialize: (mtfa-ph-serialize ds) => serializes the ds in a bytevector
- mtfa-ph-deserialize: (mtfa-ph-deserialize) => deserializes a bytevector to the ds
- mtfa-ph?: (mtfa-ph? ds) => it is the ds a ph ?
- mtfa-su-make: (mtfa-su-make) => makes a ds able to store a set. as bitmap.
- mtfa-su-reset: (mtfa-su-reset su) => clean the data set
- mtfa-su-size: (mtfa-su-size su) => returns the number of bits occupied by te su
- mtfa-su-count: (mtfa-su-count su) => returns the number of items set at true or false in the su
- mtfa-su-set-all: (mtfa-su-set-all su) => sets to true all the bits of the su
- mtfa-su-flip-all: (mtfa-su-flip-all su) => reverse all the bits of the su
- mtfa-su-set: (mtfa-su-set su index #t/#f) => sets to #t or #f the index-th bit of the su
- mtfa-su-flip: (mtfa-su-flip su index) => reverse the index-th bit of the su
- mtfa-su-get: (mtfa-su-get su index) => returns the value of the index-th bit of the su
- mtfa-su-assign: (mtfa-su-assign su su1) => assigns su1 to su
- mtfa-su-compare: (mtfa-su-compare su1 su2) => compares two sus
- mtfa-su-or: (mtfa-su-or su1 su2) => makes the or of two sus

- mtfa-su-or-equal: (mtfa-su-or-equal su1 su2) => assigns to su1 the result of the or
- mtfa-su-and: (mtfa-su-and su1 su2) => makes the and
- mtfa-su-and-equal: (mtfa-su-and-equal su1 su2) => assigns to su1 the and
- mtfa-su-xor: (mtfa-su-xor su1 su2) => makes the xor
- mtfa-su-xor-equal: (mtfa-su-xor-equal su1 su2) => assigns to su1 the xor
- mtfa-su-diff: (mtfa-su-diff su1 su2) => makes the set difference
- mtfa-su-diff-equal: (mtfa-su-diff-equal su1 su2) => assigns to su1 the set difference
- mtfa-su-get-all: (mtfa-su-get-all su) => returns all the bit set in the su
- mtfa-su-to-string: (mtfa-su-to-string su) => convert su to a string of 0 and 1
- mtfa-su-serialize: (mtfa-su-serialize su) => serializes the su in a bytevector
- mtfa-su-deserialize: (mtfa-su-deserialize bv) => deserializes a byte vector into a su
- mtfa-su-resize: (mtfa-su-resize su size) => change the size of a su
- mtfa-su-clone: (mtfa-su-clone su) => clones the su
- mtfa-su?: (mtfa-su? su) => it is really an su?
- mtfa-db-mysql-open: (mtfa-db-mysql-open) => returns the pointer to a mysqldb, need to connect using mysql-connect.
- mtfa-db-mysql-connect: (mtfa-db-mysql-connect db ip user passwd name port) => connects the db to a real one
- mtfa-db-mysql-close: (mtfa-db-mysql-close db) => releases memory
- mtfa-db-mysql-disconnect: (mtfa-db-mysql-disconnect db) => closes the connection to the db
- mtfa-db-mysql-start-transaction: (mtfa-db-mysql-start-transaction db) => it is the BEGIN TRANSACTION
- mtfa-db-mysql-rollback: (mtfa-db-mysql-rollback db) => it is the ROLLBACK
- mtfa-db-mysql-commit: (mtfa-db-mysql-commit db) => it is the COMMIT
- mtfa-db-mysql-do-sql: (mtfa-db-mysql-do-sql sd <sql statement>) => executes the sql statement on the db
- mtfa-db-mysql-last-id : (mtfa-db-mysql-last-id db) => returns the index of the last insert done on the db
- mtfa-db-mysql-p: (mtfa-db-mysql-p db) => do it is a db?
- mtfa-rand: (mtfa-rand) => uniform 64 bit random generator
- mtfa-rand-d: (mtfa-rand-d) => uniform 0.0-1.0 random generator
- mtfa-rand-ui: (mtfa-rand-ui) => uniform 32 bit random generator
- mtfa-rand-string: (mtfa-rand-string size) => returns a string of size digits
- mtfa-rand-alfanum: (mtfa-rand-alfanum size alphabet) => returns a string of length size made by the characters in the alphabet
- mtfa-rand-seed: (mtfa-rand-seed number) => seeds the above random generators
- mtfa-rand-mat: (mtfa-rand-mat) => 64 bit more strong random generator
- mtfa-rand-mat-ui: (mtfa-rand-mat-ui) => 32 bit more strong random generator
- mtfa-rand-mat-d: (mtfa-rand-mat-d) => 0.0-1.0 more strong random generator
- mtfa-rand-mat-seed: (mtfa-rand-mat-seed number) => seeds the above rangom generators

- mtfa-b64-encode: (mtfa-b64-encode string/byte vector) => base64 encoder
- mtfa-b64-decode-s: (mtfa-b64-decode-s str) => base64 decoder to a string
- mtfa-b64-decode-bv: (mtfa-b64-decode-bv str) => base64 decoder to a bytevector
- mtfa-b64-url-encode: (mtfa-b64-url-encode str/bv) => encode in url base 64
- mtfa-b64-url-decode-s: (mtfa-b64-url-decode-s str) => decode url base64 to string
- mtfa-b64-url-decode-bv: (mtfa-b64-url-decode-bv str) => decode url base64 to a byte vector
- mtfa-send-mail: (mtfa-send-mail smtpserver username password from params msg personal_ca_path check_server_cert_and_or_hostname) //0 no, 1 check only server, 2 check only hostname, 3 check server and hostname) => sends an email
- mtfa-send-mail-prepare-annexes: (mtfa-send-mail-prepare-annexes domain body-text list-of-name-type => example: (mtfa-send-mail-prepare-annexes "server domain" "body text of the email" (list (list "image.png" "pdf" (mtfa-b64-encode (fs-io-to-bv "/tmp/image.png")) #t) (list "" "text" "Hi, am I" #t)))
- mtfa-base62: (mtfa-base62 data) => makes the base 62 of the data
- mtfa-debase62: (mtfa-debase62 str) => debase from base62
- mtfa-micros: (mtfa-micros) => returns the current microseconds
- CompilePattern: (CompilePattern search-pattern) => copile a regex pattern
- FindPattern: (FindPattern regex string) => find regex in the string (#t/#f)
- FindCompiledPattern: (FindCompiledPattern <compiled regex> string) => finds compiled pattern in the string
- ReplaceAll : (ReplaceAll string pattern subst) => replaces all occurrences of the pattern in the string with subst
- Show: (Show params...) => print current date and the values of all the parameters
- Show!: (Show! params...) => print the values of all the parameters
- Show!eol: (Show!eol params...) => print the values of all the parameters. Do not add EOL
- mtfa-sqlite3-do: (mtfa-sqlite3-do <name> <sql>) => executes the sql statement on the sqlite3 file db named <name>
- mtfa-sqlite3-open: (mtfa-sqlite3-open <name>) => opens the sqlite3 db from file <name>
- mtfa-sqlite3-close: (mtfa-sqlite3-close db) => closes the sqlite3 connection
- mtfa-sqlite3-execute: (mtfa-sqlite3-execute db sql) => executes the sql statement on the opened db
- mtfa-dispatcher::make: (mtfa-dispatcher::make <optional not-found function) => create a functional dispatcher where you can add, delete, find a function by key, possibly passing a function to be executed when the key is not found
- mtfa-dispatcher::call: (mtfa-dispatcher::call <disp> <fun key> <parameters>) => call the function associated to the key in the dispatcher passing the parameters
- mtfa-dispatcher::add: (mtfa-dispatcher::add disp key fun) => associates in the dispatcher disp the function fun to the key
- mtfa-dispatcher::del: (mtfa-dispatcher::del disp key) => remove from the dispatcher the function associated to the key
- mtfa-strong-random: (mtfa-strong-random <num of bits>) => makes a strong random number of <num bits> bits
- mtfa-make-rsa: (mtfa-make-rsa) => makes an rsa generator

- mtfa-rsa-make-key: (mtfa-rsa-make-key rsa size) => makes the keys of size bits using the rsa generator rsa
- mtfa-rsa-pub-cipher: (mtfa-rsa-pub-cipher rsa string/bv padding) => rsa public cyphering with padding (0, 1, 2)
- mtfa-rsa-priv-decipher: (mtfa-rsa-priv-decipher rsa bv padding) => deciphers an rsa pub ciphered message. Gives a bytevector
- mtfa-rsa-priv-cipher: (mtfa-rsa-priv-cipher rsa string/bv 0) => rsa private cyphering with mandatory padding 0
- mtfa-rsa-pub-decipher: (mtfa-rsa-pub-decipher rsa bv 0) => deciphers a rsa private cipher giving a bytevector
- mtfa-rsa-sign: (mtfa-rsa-sign rsa message) => gives the signed message (as bytevector)
- mtfa-rsa-verify: (mtfa-rsa-verify rsa signed original) => example: (mtfa-rsa-verify r (mtfa-rsa-sig r "Hello") "Hello"), return #t/#f
- mtfa-rsa-make-key-p-q: (mtfa-rsa-make-key-p-q rsa p q) => passes two primes for making the rsa keys
- mtfa-rsa-get-pub-key: (mtfa-rsa-get-pub-key rsa) => returns the pub key
- mtfa-rsa-get-priv-key: (mtfa-rsa-get-priv-key rsa) => returns the priv key
- mtfa-rsa-set-pub-key: (mtfa-rsa-set-pub-key rsa pub-key) => sets the pub key
- mtfa-rsa-set-priv-key: (mtfa-rsa-set-priv-key rsa priv-key) => sets the private key
- mtfa-hash: (mtfa-hash index message) => makes the hash of the message. The indexes are ("SHA1" . 0) ("SHA224" . 1) ("SHA256" . 2) ("SHA384" . 3) ("SHA512" . 4) ("MD5" . 5) ("RIPEMD160" . 6)
- mtfa-hash-hs: (mtfa-hash-hs index message) => as mtfa-hash but returns a hex string
- mtfa-hash-b64: (mtfa-hash-b64 index message) => same, returns base64 string
- mtfa-hmac: (mtfa-hmac index key message) => returns the hmac of a message. ("SHA1" . 0) ("SHA224" . 1) ("SHA256" . 2) ("SHA384" . 3) ("SHA512" . 4) ("MD5" . 5) ("RIPEMD160" . 6)
- mtfa-hmac-hs: (mtfa-hmac-hs index key message) => as mtfa-hcam but returns an hex string
- mtfa-hmac-b64: (mtfa-hmac-b64 index key message) => same, returns base64 string
- mtfa-string-split-regex: (mtfa-string-split-regex string regex <optional separator>) => example: (mtfa-string-split-regex "one two three four" "o") => ("" "ne tw" " three f" "ur") (mtfa-string-split-regex "one two three four" "o" #\r) => ("" "ne tw" " th" "ee f" "u" "")
- mtfa-bv-to-hex-string: (mtfa-bv-to-hex-string bv) => converts the bytevector to a hex string
- mtfa-hex-string-to-bv: (mtfa-hex-string-to-bv hs) => converts the hex string in a byte vector
- mtfa-cifra-aes-js-hex : (mtfa-cifra-aes-js-hex msg key) => ciphers the mesg giving an hex string
- mtfa-decifra-aes-js-hex-s: (mtfa-decifra-aes-js-hex-s msg key) => deciphers the ciphered message
- mtfa-decifra-aes-js-hex-bv: (mtfa-decifra-aes-js-hex-bv msg key) => same but gives a byte vector
- mtfa-cifra-aes-js: (mtfa-cifra-aes-js msg key) => ciphers and gives a bytevector
- mtfa-decifra-aes-js: (mtfa-decifra-aes-js msg key) => deciphers giving a bytevector
- mtfa-cifra-aes-js-b64 : (mtfa-cifra-aes-js-b64 msg key) => ciphers in base64
- mtfa-decifra-aes-js-b64-s : (mtfa-decifra-aes-js-b64-s msg key) => deciphers from b64 to string

- mtfa-decifra-aes-js-b64-bv : (mtfa-decifra-aes-js-b64-bv msg key) => deciphers from base 64 to bytevector
- mtfa-bv-xor: (mtfa-bv-xor bv1 bv2) => trivial
- mtfa-bv-or: (mtfa-bv-or bv1 bv2) => trivial
- mtfa-bv-and: (mtfa-bv-and bv1 bv2) => trivial
- mtfa-base32-encode: (mtfa-base32-encode msg) => base32 encoding
- mtfa-base32-encode-bv: (mtfa-base32-encode-bv msg) => base32 encoding
- mtfa-base32-encode-s: (mtfa-base32-encode-s msg) => base32 encoding
- mtfa-base32-decode: (mtfa-base32-decode msg) => base32 decoding
- mtfa-base32-decode-bv: (mtfa-base32-decode-bv msg) => base32 decoding
- mtfa-base32-decode-s: (mtfa-base32-decode-s msg) => base32 decoding
- mtfa-seconds: (mtfa-seconds) => number of seconds as unix time
- mtfa-left-pad: (mtfa-left-pad string size padding) => example: (mtfa-left-pad "one" 30 #\p) => "pppppppppppppppppppppppppppone"
- mtfa-totp-gen: (mtfa-totp-gen secret length time-offset period) => OTP generator
- mtfa-star: mtfa-star-make, mtfa-star?, mtfa-star-insert, mtfa-star-search, mtfa-star-build, mtfa-star-get-rules, mtfa-star-serialize, mtfa-star-deserialize) =>Example:

```
(define st (mtfa-star-make))
(mtfa-star-insert st '(1 2 3 4) 10 0)     ;;0: list of num, id, jolly
(mtfa-star-insert st '(2 3 4 5) 11 0)     ;;1
(mtfa-star-insert st '(3 4 0 6 7) 12 0)   ;;2
(mtfa-star-insert st '(3 4 1 6 7) 13 0)   ;;3
(mtfa-star-insert st '(3 4 99 6 7) 14 0)  ;;4
(mtfa-star-insert st '(3 0 99 6 7) 15 0)  ;;5
;;note that the 13 and 14 are equal to 12 given the jolly (0 in this case)
(mtfa-star-build st) ;;the mandatory build
(mtfa-star-get-rules st 0) => (10)
(mtfa-star-get-rules st 1) => (11)
(mtfa-star-get-rules st 2) => (13 12) ;;the 13 matches the 12 because of the jolly
(mtfa-star-get-rules st 3) => (15 14 12) ;;the 14 matches the 12 and the 15 because of the
jolly
(mtfa-star-get-rules st 4) => (12)
(mtfa-star-get-rules st 4) => (15)
;;
;;now searching
(mtfa-star-search st '(3 4 99 6 7)) => 3
(mtfa-star-search st '(3 5 99 6 7)) => 5
(mtfa-star-search st '(3 6 99 6 7)) => 5
(mtfa-star-search st '(3 4 1 6 7))  => 2
```

- mtfa-compress: (mtfa-compress msg level) => trivial
- mtfa-uncompress: (mtfa-uncompress msg) => trivial
- mtfa-http-compress: (mtfa-http-compress msg level) => as compress but for http communications
- mtfa-http-uncompress: (mtfa-http-uncompress msg) => as uncompress but for http communications
- mtfa-cookie-make: (mtfa-cookie-make key value expiration howmanysecs path domain http_only secure) => trivial
- mtfa-compile-pattern: (mtfa-compile-pattern regexp) => compile pattern using new libraries for regular expression

- mtfa-find-all-matching-positions: Example
  (mtfa-find-all-matching-positions "put the book on the table, close the door" (mtfa-compile-patter
  "o+")) => ((9 . 2) (13 . 1) (29 . 1) (38 . 2))

- mtfa-replace-all: Example
  (mtfa-replace-all (mtfa-compile-pattern "o+" #t) "put the book on the table, close the door"
  "II" #t) => "put the bIIk IIn the table, clIIse the dIIr"

- mtfa-blum-make, mtfa-blum-get-values, mtfa-blum-get-next, mtfa-blum-get-prev, mtfa-blum-get-next-i
  mtfa-blum-get-prev-ith, mtfa-blum-get-next-ith-slow, mtfa-blum-set-values, mtfa-bv-to-num,
  mtfa-num-to-bv, mtfa-blum-make-token, mtfa-blum-identify-token: => the class of blum numbers,
  trivial

- mtfa-fuzzy-find: check for similarty. Example:
  (mtfa-fuzzy-find "Hello dolly" "Hallo dely" 2) => #f (not found)
  (mtfa-fuzzy-find "Hello dolly" "Hallo dely" 4) => 3 (found with 3 insert/delete/change)

- mtfa-sss-make, mtfa-sss?, mtfa-sss-insert, mtfa-sss-search, mtfa-sss-unlink, mtfa-sss-getval,
  mtfa-sss-getall, mtfa-sss-getallid, mtfa-sss-ls2bv, mtfa-sss-serialize, mtfa-sss-deserialize:
  examples:
  (define ds (mtfa-sss-make))
  (mtfa-sss-insert ds '("one" "two" "three"))
  (mtfa-sss-insert ds '("4" "5" "6" "7"))
  (mtfa-sss-insert ds '("one" "four" "five"))
  (mtfa-sss-insert ds '("4" "5" "999" "seventy" "fifty" "ninety" "eleven"))
  (mtfa-sss-getall ds '()) => (("one" "two" "three") ("one" "four" "five") ("4" "5" "6" "7")
  ("4" "5" "999" "seventy" "fifty" "ninety" "eleven"))
  (mtfa-sss-getall ds '("one")) => (("one" "two" "three") ("one" "four" "five"))
  (mtfa-sss-getall ds '("4" "5")) => (("4" "5" "6" "7") ("4" "5" "999" "seventy" "fifty" "ninety"
  "eleven"))
  (mtfa-sss-getall ds '("4" "5" "999" "seventy" "fifty" "ninety")) => (("4" "5" "999" "seventy"
  "fifty" "ninety" "eleven"))
  The sss structure can manage complex multidimensional set of data

- mtfa-hash-keccak: (mtfa-hash-keccak bv) => makes the keccak hash of a bytevector

- mtfa-eth-make-priv-key, mtfa-eth-get-pub-key, mtfa-eth-get-pub-key-serialized, mtfa-eth-serialize-
  mtfa-eth-sign-msg => the ethereum functions to manage wallet and private keys.

- mtfa-morton-16-encode, mtfa-morton-16-decode, mtfa-morton-32-encode, mtfa-morton-32-decode
  => morton number encoding and decoding. To index 2D spatial data structures.

- mtfa-make-cimg,mtfa-cimg-p, mtfa-cimg-set-space, mtfa-cimg-set-linewidth, mtfa-cimg-erase,
  mtfa-cimg-flush, mtfa-cimg-close, mtfa-cimg-open, mtfa-cimg-set-linecolor-string, mtfa-cimg-set-li
  mtfa-cimg-set-fillcolor-string, mtfa-cimg-set-fillcolor, mtfa-cimg-set-filltype, mtfa-cimg-set-bg-
  mtfa-cimg-set-bg, mtfa-cimg-draw-box, mtfa-cimg-draw-label, mtfa-cimg-move, mtfa-cimg-draw-circle,
  mtfa-cimg-draw-line, mtfa-cimg-draw-point, mtfa-cimg-font-name, mtfa-cimg-font-size, mtfa-cimg-tex
  mtfa-cimg-image => access to the CIMG graphical library

- mtfa-onvif-make, mtfa-onvif?, mtfa-onvif-connect, mtfa-onvif-move => the onvif library to
  control remote camera over IP. Example:
  (receive (ok onv)
  (apply mtfa-onvif-make '("http://127.0.0.1:30001/onvif" "admin" "password"))
  (mtfa-onvif-connect onv)
  (mtfa-onvif-move onv 0.195 -0.48 0 1) ;;x, y, zoom, speed
  ...)

- define-syntax-public: to define public macros in Guile scheme

- string-case: the string-case operator. Example: (string-case str ("Aaa" (Show! 1)) ("aaa"
  (Show! 2)) ("ddd" (Show! 3)) (else (Show! 100)))

- define-class-public: to define public classes

- define-method-public: to define public methods

- define-method*: to define methods with optional parameters
- define-method*-public: as here above, but public
- repeat: the repeat operator. Usage (repeat 100 expr expr ...)
- collect: the collect operator accumulates in a list the execution of s-expressions. Usage: (collect 1000 expr expt ...) => a list of values
- defun, defun-public, defun*, defune*-public: extends the define operator adding the return and error control. Examples:
  (defun funct (par1 par2) (if (null? par1) (return #f)) (/ par1 par2)): return #f if par1 is null, otherwise returns the division of par1 by par2.
  (defun funct (par1 par2) => #f (if (null? par1) (return #f)) (/ par1 par2)): return #f if par1 is null, otherwise returns the division of par1 by par2. If there is an error (i.e: par2 is 0), catches the error and returns #f.
- mtfa-get-pattern (str pat): Finds a pattern in a string and returns it
- mtfa-typeof (v): Gives the type of a scheme variable
- mtfa-set-subsets (s) build the list of all subsets of the list/set s
- mtfa-memoize (fun): memoizes the function, speeding up the run.
  Example: (defun fib (n) (if (< n 2) 1 (+ n (fib (- n 1)) (fib (- n 2)))))
  ,time (fib 40) => 28.605713s real time !!!too much time
  (define fib (mtfa-memoize fib))
  ,time (fib 1000) => 0.002019s real time !!!very fast, now
- the amb non deterministic operator
- mtfa-char-set-alfa-lower, mtfa-char-set-alfa-upper, mtfa-char-set-alfa, mtfa-char-set-alfanum-lowe
  mtfa-char-set-alfanum-upper,mtfa-char-set-alfanum, mtfa-char-set-hex-lower, mtfa-char-set-hex-uppe
  mtfa-char-set-all-bv, mtfa-char-set-all, mtfa-char-set-half-bv, mtfa-char-set-half: many different char sets
- (mtfa-time-stamp-make duration-in-secs), (mtfa-time-stamp-check ts): make a checkable timestamp using hmac
- (mtfa-entropy bv): measures the entropy of a bytevector
- mtfa-string->bytevector (s), mtfa-bytevector->string (s): converts a string into a bytevector and viceversa
- (is expr) (isnt expr): infix notation for s.expressions
- mtfa-define-public-symbol (s ...): defines public symbols as themselves. Example:
  (mtfa-define-public-symbol newsym)
  newsym => newsym
- (mtfa-define-public-symbol-value prefix (symb . 10)): makes the new symbol prefix::symb assigning the value of 10
- mtfa-define-public-symbol-value-from-assoc-list (sprefix assoc-list): defines a set of new symbols as sprefix::symbol from assoc list and assign to them a value
- mtfa-define-symbol-value-pairs-from-json (sprefix filename): as the previous but associations are read from the file in json format
- (json-let '(("a" . 1)("b" . 2)) (a b) (Show! a ", " b))
- json-make:
  (define a 10)
  (define b 20)
  (json-make a b) => ((a . 10) (b . 10))

- mtfa-record-make: macro that make a record. Example:
  (mtfa-record-make myrecord field1 field2) creates the functions
  myrecord?, myrecord-field1!, myrecord-from-json, myrecord-to-json, myrecord-clone, myrecord-field2
  myrecord-from-json-string, myrecord-to-json-string, myrecord-field1, myrecord-field2!, myrecord-mal
- (mtfa-make-closure funname prefix (par1 par2) ((a par1)(b par2)) ((add n)(+ n a b))((mul
  n)(* n a b))): translate it in:
  (define (funname par1 par2) (let ((a par1)(b par2))
  (lambda (cmd . pars)
  (match (cons cmd pars)
  (('prefix::add n) (+ n a b))
  (('prefix::mul n) (* n a b)))))
  (mtfa-define-public-symbol prefix::add)
  (mtfa-define-public-symbol prefix::mul)
  After that you have run (mtfa-make-closure funname prefix (par1 par2) ((a par1)(b par2))
  ((add n)(+ n a b))((mul n)(* n a b))), you can:
  (define fun (funname 100 200))
  and then
  (fun prefix::add 1) => 301
- (mtfa-asq-make), (mtfa-asq-enq! q elt), (mtfa-asq-deq! q), (mtfa-asq-top! q), (mtfa-asq-empty?
  q), (mtfa-asq-len q): an asynchronous queue
- (mtfa-logging-run filename #:optional (not-on-screen '(WARN INFO))), (mtfa-logging-shutdown)
  "Creates the logger, writing on <filename> and tells to not report to video the logs marked
  'WARN and 'INFO
  Example: (mtfa-logging-run <name of the log file> #:not-on-screen '(WARN INFO))"

3.2 module error-handler

- (try-catch result-in-case-of-error expr ...): catch errors and shows the error string and
  the line number
- (mtfa-noerr result-in-case-of-error expr ...): try-catch without showing the error string

3.3 module serializer

- (mtfa-serializer obj), (mtfa-deserializer obj), (mtfa-serializer-b64 obj)(mtfa-deserializer-b64
  obj)
  serializes and deserializes almost any object, included the mtfa structures

3.4 module unordered-set

- mtfa-unordered-set: a class for managing unordered sets. persistent.
  Methods:
  (mtfa-us::add (ds <mtfa-unordered-set>) key)
  (mtfa-us::del (ds <mtfa-unordered-set>) key)
  (mtfa-us::for-each (ds <mtfa-unordered-set>) f_1)
  (mtfa-us::map (ds <mtfa-unordered-set>) f_1)
  (mtfa-us::fold (ds <mtfa-unordered-set>) f_2 prec)
  (mtfa-us::restore (ds <mtfa-unordered-set>))

  Example:
  (define us (make <mtfa-unordered-set> #:fsave "persist.dat")) ;;if fsave is not nil, the
  persistence is activated
  ;;the methods: mtfa-us::add, mtfa-us::del, mtfa-us::fold, mtfa-us::for-each, mtfa-us::map,
  mtfa-us::restore
  (mtfa-us::add us "a key") ;;adds a key
  (mtfa-us::add us "a second key") ;;adds another key
  (mtfa-us::map us (lambda (k) (Show! k) k)) ;;shows the keys and return their list

```
a key
a second key
("a second key" "a key")
When you restarts the classe (make...) the previous data are automatically recovered (no
need to call restore).
i.e: the persistent files in this example,are:
 persist.dat.mtfa.us
 persist.dat.mtfa.us-shm
 persist.dat.mtfa.us-wal
 The persistence uses a sqlite3 as archive
```

3.5 module unordered-map

- mtfa-unordered-map: a class for managing unordered maps. persistent
  Similar to the unordered-set.
  (define our-umap (make <mtfa-unordered-map> #:fsave "persist.dat"))
  The methods are:
  (mtfa-um::add (ds <mtfa-unordered-map>) key value)
  (mtfa-um::get (ds <mtfa-unordered-map>) key)
  (mtfa-um::del (ds <mtfa-unordered-map>) key)
  (mtfa-um::for-each (ds <mtfa-unordered-map>) f_1)
  (mtfa-um::map (ds <mtfa-unordered-map>) f_1)
  (mtfa-um::fold (ds <mtfa-unordered-map>) f_2 prec)
  (mtfa-um::restore (ds <mtfa-unordered-map>))

3.6 module simpledb

This module contains functions to access mysql and sqlite3 rdbms. Furthermore a set of special forms (macro) build complete db schema from a database and declare all the tables and attributes inside the guile scheme environment. Really useful when manipulate db. Some examples will explain better.

- MYSQL (it is supposed that you know mysql):
  (db-interface::set-db-coordinates ip user password name port)
  (db-interface::DoSqlQuery db-pars query #:optional lastid)
  (db-interface::DoConnect db-pars)
  (db-interface::DoDisconnect db)
  (db-interface::StartSqlSession db-pars)
  (db-interface::StopSqlSession db)
  (db-interface::CommitSqlSession db)
  (db-interface::RollbackSqlSession db)
  (db-interface::DoSqlCommitted db query #:optional lastid)
  (db-interface::get-lastid db)

- SQLITE3 (it is supposed that you know sqlite3):
  (db-interface::Sqlite3::Do connector sql)
  (db-interface::Sqlite3::Open connector)
  (db-interface::Sqlite3::Close connector)

- MYSQL:
  (SQL-BuildDbSchema db-connector)
  (SQL-Select connector SELECT attrs-list FROM from-list WHERE where-clauses-list show-sql-string)
  (SQL-Select connector SELECT attrs-list FROM from-list WHERE where-clauses-list)
  (SQL-Select connector SELECT attrs-list FROM from-list show-sql-string)
  (SQL-Select connector SELECT attrs-list FROM from-list)

  (SQL-Update connector UPDATE table-name SET attrs-list TO values-list WHERE where-clauses-list
  show-sql-string)

```
(SQL-Update connector UPDATE table-name SET attrs-list TO values-list WHERE where-clauses-list)
(SQL-Update connector UPDATE table-name SET attrs-list TO values-list  show-sql-string)
(SQL-Update connector UPDATE table-name SET attrs-list TO values-list)
(SQL-Insert connector INSERT INTO table-name attrs-list VALUES values-list show-sql-string
return-lastid)
(SQL-Insert connector INSERT INTO table-name attrs-list VALUES values-list return-lastid)
(SQL-Insert connector INSERT INTO table-name attrs-list VALUES values-list)

SQLITE3:
(SQL3-BuildDbSchema db-name)
(SQL3-Select connector SELECT attrs-list FROM from-list WHERE where-clauses-list show-sql-string)
(SQL3-Select connector SELECT attrs-list FROM from-list WHERE where-clauses-list)
(SQL3-Select connector SELECT attrs-list FROM from-list show-sql-string)
(SQL3-Select connector SELECT attrs-list FROM from-list)

(SQL3-Update connector UPDATE table-name SET attrs-list TO values-list WHERE where-clauses-list
show-sql-string)
(SQL3-Update connector UPDATE table-name SET attrs-list TO values-list WHERE where-clauses-list)
(SQL3-Update connector UPDATE table-name SET attrs-list TO values-list  show-sql-string)
(SQL3-Update connector UPDATE table-name SET attrs-list TO values-list)
(SQL3-Insert connector INSERT INTO table-name attrs-list VALUES values-list show-sql-string)
(SQL3-Insert connector INSERT INTO table-name attrs-list VALUES values-list)


Example with a sqlite3 db. The file name of the db is "mydb"
the db contain:
CREATE TABLE contacts (
contact_id INTEGER PRIMARY KEY,
first_name TEXT NOT NULL,
last_name TEXT NOT NULL,
email TEXT NOT NULL UNIQUE,
phone TEXT NOT NULL UNIQUE
);
INSERT INTO contacts VALUES(1,'john','doyle','jd@mycompanycom','(234) 235-5678');
INSERT INTO contacts VALUES(2,'emily','nettle','en@mycompanycom','(234) 235-5679');
INSERT INTO contacts VALUES(3,'july','cradle','jc@mycompanycom','(234) 235-5670');
COMMIT;

Now, let's see how to manipulate using these special forms.
(SQL3-BuildDbSchema "mydb")
;;the internal schema has been created. now under the prefix DB: there is a DB name "main"
and the tables, and the attributes.
(SQL3-Select "mydb" SELECT (DB::main.contacts.contact_id) FROM (DB::main.contacts)) => (("3")
("1") ("2"))
```

3.7 module eis

This module contains functions to interact with to access manager, validation manager and communication manager.

- eis::http-content-type-html
  eis::http-content-type-text
  eis::http-content-type-b64
  eis::http-content-type-xml
  eis::http-content-type-javascript
  eis::http-answer-ok
  eis::http-content-length
  eis::http-connection-close
  eis::http-connection-keep-alive

```
eis::http-eoln
eis::http-end-of-headers
eis::BaseLib::BuildHTTPAnswer
eis::GiveHTTPAnswer
eis::GiveHTTPHtmlAnswer
eis::GiveHTTPXmlAnswer
eis::GiveHTTPTextAnswer
eis::GiveHTTPB64Answer
eis::GiveHTTPJSONAnswer
eis::GiveErrorHTTP404
eis::GiveErrorHTTP401
eis::GiveErrorHTTP500
eis::LoadPage
eis::SplitPostData
mtfa-eis-get-current-query
mtfa-eis-get-value-current-query
mtfa-eis-get-value-current-headers
mtfa-eis-get-current-headers
mtfa-eis-get-value-current-cookies
mtfa-eis-get-current-body
mtfa-eis-get-current-method
mtfa-eis-get-current-protocol
mtfa-eis-get-current-host
mtfa-eis-get-current-port
mtfa-eis-get-current-pars
mtfa-eis-get-current-tag
mtfa-eis-get-current-url
mtfa-eis-get-current-uri
mtfa-eis-get-current-user-passwd
mtfa-eis-get-current-mac-src
mtfa-eis-get-current-ip-src
mtfa-eis-get-current-ip-dst
mtfa-eis-get-current-port-src
mtfa-eis-get-current-port-dst
mtfa-eis-get-current-multipart
mtfa-eis-get-current-ssl-servername
mtfa-eis-get-ip-network-protocol
mtfa-eis-set-ip-real-data-size
mtfa-eis-get-ip-data-len
mtfa-eis-get-ip-id-probe
mtfa-eis-get-raw-data
mtfa-eis-put-raw-data
(eis::function-pointer-add k v)
(eis::function-pointer-get k)
(eis::BaseLib::ContinueDoNotClose)
(eis::BaseLib::Close)
(eis::BaseLib::ContinueAfterAccept)
(eis::BaseLib::NeedMoreData)
(eis::BaseLib::BlockSession . other)
(eis::GiveAnswerAndCloseAll answer)
(eis::GiveAnswerAndNotClose answer)
(eis::Redirect where #:optional (headers ""))
(eis::GiveFileAndClose http-answer headers filename)
(eis::GiveFileAndNotClose http-answer headers filename)
eis::BaseLib::PassToServer
(define eis::http-content-type-html "Content-Type: text/html; charset=utf-8")
(define eis::http-content-type-text "Content-Type: text/plain; charset=UTF-8")
(define eis::http-content-type-b64  "Content-Type: text/plain; charset=UTF-8\r\nContent-transfer-e
base64")
(define eis::http-content-type-xml "Content-Type: text/xml; charset=utf-8")
(define eis::http-content-type-json "Content-Type: application/json; charset=utf-8")
```

```
(define eis::http-content-type-javascript "Content-Type: application/javascript; charset=utf-8")
(define-public eis::http-content-type-css "Content-Type: text/css; charset=utf-8")
(define eis::http-answer-ok "HTTP/1.1 200 OK")
(define (eis::http-content-length l)
  (string-append "Content-Length: " (if (number? l) (number->string l) l)))
(define eis::http-connection-close "Connection: close")
(define eis::http-connection-keep-alive "Connection: keep-alive")
(define eis::http-eoln "\r\n")
(define eis::http-end-of-headers "\r\n\r\n")
(eis::BaseLib::BuildHTTPAnswer http-answer headers cookies body)
(eis::GiveHTTPAnswer http-answer headers cookies body #:optional (doclose #t))
(eis::GiveHTTPHtmlAnswer page #:optional (doclose #t) #:key
  (answer eis::http-answer-ok)
  (headers (string-append eis::http-content-type-html eis::http-eoln eis::http-connection-close
eis::http-eoln))
  (cookies "")
  (zip #f)
  )
(eis::GiveHTTPXmlAnswer page  #:optional (doclose #t))
(eis::GiveHTTPTextAnswer page  #:optional (doclose #t))
(eis::GiveHTTPB64Answer page  #:optional (doclose #t))
(eis::GiveHTTPJSONAnswer data #:optional (doclose #t))
(eis::GiveErrorHTTP404  #:optional (doclose #t))
(eis::GiveErrorHTTP401  #:optional (doclose #t))
(eis::GiveErrorHTTP500  #:optional (msg "The request cannot be satisfied due to an internal
server error.") (doclose #t))
(eis::SplitPostData post-data)
mtfa-eis-content-type-for-unknown-file (filename)
(eis::GiveHttpMimeAnswer http-answer headers cookies body #:optional (doclose #t))
```

The meaning of these definitions and functions is really simple, for a WEB programmer. Anyway, in the following, a simple example is given to show how to use these functions.

Firstly, a simple ARS file containing at least the following code that for each http communication pointing to the url in the set usr_service

```
define urlset url_service = { /msvc };
DEFINE AR "AR-MICROSVC"
      CONDITION
           http.url is in url_service
      ACTION
           MANAGE "MICROSVC"
;
```

Then have a look to the scheme code, whose file name is declared in the configuration file of the VAPP-Virtual Application (LbUser_LspCode=...)

```
;;START SCHEME CODE
(use-modules
 (mtfa error-handler)
 (mtfa utils)
 (mtfa serializer)
 (mtfa unordered-set)
 (mtfa unordered-map)
 (mtfa star-map)
 (mtfa simple_db)
 (mtfa certs)
 (mtfa eis)
 (mtfa va)
 (mtfa extset)
 (mtfa umset)
 (mtfa web)
```

```
 (mtfa brg)
 (mtfa lazy-seq)
 (mtfa domain-fiber-server)
 ;;
 (pfds sets)
 (simple-zmq)
 ;;
;; (gnutls)
 ;;
 ;;((rnrs records syntactic) #:prefix rnrs::)
 (rnrs bytevectors)
 (rnrs arithmetic bitwise)
 ((rnrs io ports)
  #:select (string->bytevector bytevector->string)
  #:prefix ioports:)
 ;;
 (srfi srfi-1)
 (srfi srfi-9)
 (srfi srfi-11)
 ((srfi srfi-18)
  #:prefix srfi-18::)
 (srfi srfi-19)
 (srfi srfi-26)
 ;;(srfi srfi-28)
 (srfi srfi-43)
 (srfi srfi-60)
 (web uri)
 (system foreign)
;;
 (ice-9 format)
 (ice-9 ftw)
 (ice-9 rdelim)
 (ice-9 pretty-print)
 (ice-9 regex)
 (ice-9 iconv)
 (ice-9 string-fun)
 (ice-9 peg)
 (ice-9 peg string-peg)
 (ice-9 vlist)
 (ice-9 q)
 (ice-9 binary-ports)
 (ice-9 threads)
 (ice-9 hash-table)
 (ice-9 control)
 (ice-9 match)
 (ice-9 receive)
 (ice-9 eval-string)
 (ice-9 arrays)
 ;;
 (oop goops)
 (oop goops describe)
 ;; (sxml simple)
 ;; (sxml ssax)
 ;; (sxml xpath)
 (json)
 (system syntax)
 (system foreign)
 ;;
 (web client)
 ;;
 (ffi blis)
```

```
 )

;;
;;Init random
(mtfa-rand-seed (mtfa-micros))
;;
;;Initialization terminated. Program started
(defun-public arCheckRole (li pbuf)
  (try-catch #t
    (Show "Received: " (map (cut bytevector->string <> "UTF-8") li))
    (Show "SQL: " (db-interface::DoSqlQuery '("127.0.0.1" "username" "password" "db name"
3306) "show tables"))
    #t))

;;Hooking function
;;Activation example: GET "http://127.0.0.1:9999/msvc?name=john&service=access"
(defun Manage::MSCV (actionl pbuf)
  (eis::GiveErrorHTTP401) =>
  (Show "actionl: " actionl ". pbuf: " pbuf)
  ;; (va::vaeng::GetFromSet )

  (Show "Host: " (mtfa-eis-get-current-host pbuf)
", " (mtfa-eis-get-current-query pbuf) ;;all the query section (both for get and post)
". Name: " (mtfa-eis-get-value-current-query pbuf "name")
". Headers: " (mtfa-eis-get-current-headers pbuf)
". Header[Host]: " (mtfa-eis-get-value-current-headers pbuf "host")
)
  ;; (write (mtfa-eis-get-current-query pbuf)) (newline)
  ;; (display (mtfa-eis-get-current-query pbuf)) (newline)

  (Show "SQL: " (db-interface::DoSqlQuery '("127.0.0.1" "username" "password" "dbname" 3306)
"show tables"))

  (eis::GiveHTTPTextAnswer (string-append "ret=OK&value=" (TimeStamp) "\n"))
  )


;;Add the HOOK
(eis::function-pointer-add "MICROSVC" Manage::MSCV)


(define (KamRun dummy1 dummy2 fun pbuf)
  ((eval-string fun) pbuf))

;;HTTP microservices example
(defun Manage::API (actionl pbuf)
  (eis::GiveErrorHTTP401) =>
  (Show "In API...")
  (eis::GiveHTTPJSONAnswer '(("answer" . "ok")))
  )

(define (LispFun1 host cookie)
  (Show host ", " cookie)
  #t)

(defun Manage::API1 (actionl pbuf)
  (eis::GiveErrorHTTP401) =>

;;(eis::BaseLib::PassToServer ip_dest porta_dest host method headers url body)

  (eis::GiveHTTPAnswer "200 OK" "MyHeader: none" "Set-Cookie: label=value" "answer=ok&code=199")
```

```
  )

;;Add HOOK
(eis::function-pointer-add "API" Manage::API)
(eis::function-pointer-add "API1" Manage::API1)


;;
;;
;;JSON microservices manager (CmdManager) example
(define (JSON-ANSWER::error msg errid)
`((success . #f)(message . ,msg) (errorid . ,errid)))
;;
(define (JSON-ANSWER::success msg data_name data)
`((success . #t)(message . ,msg) (data . ((,data_name . ,data)))))

(defun JSONAPI::add (k v pbuf set-names)
(eis::GiveHTTPJSONAnswer (JSON-ANSWER::error "General System Error" -1))  =>
(Show "Got add" v)
(let ((v1 (assoc-ref v "v1"))
  (v2 (assoc-ref v "v2")))
     (eis::GiveHTTPJSONAnswer (JSON-ANSWER::success "OK" "DATANAME" (+ v1 v2)))))

#|
Conversin JSON-SCHEME
{"k": "v"} => (k . v)
[1, 2, 3] => #(1 2 3)
ES:
{"add": {"v1": 10, "v2": 20}} => (("add" . (("v1" . 10) ("v2" . 20))))

`((success . #t)(message . ,msg) (data . ((,data_name . ,data))))
{"success": true, "message": "content of the variable msg", "data": {"cont of data_name":
"cont of data"}}
|#

;;(define CmdManager (mtfa-web-make-json-requests-manager "JSONAPI"))

;;
;;CURL: curl http://127.0.0.1:8888/ReqJson --data '{"add": {"v1": 10, "v2": 20}}' -H 'Content-Type
;;CURL: curl http://127.0.0.1:8888/ReqJson --data '{"sub": {"v1": 10, "v2": 20}}' -H 'Content-Type
;;CURL: curl http://127.0.0.1:8888/ReqJson --data '{"mul": {"v1": 10, "v2": 20}}' -H 'Content-Type
;;CURL: curl http://127.0.0.1:8888/ReqJson --data '{"div": {"v1": 10, "v2": 20}}' -H 'Content-Type

;;
;;(CmdManager 'add-handler "add" JSONAPI::add)

;;(CmdManager 'add-handler "sub" JSONAPI::sub)
;;(CmdManager 'add-handler "mul" JSONAPI::mul)
;;(CmdManager 'add-handler "div" JSONAPI::div)
;;END SCHEME CODE
```

3.8 module va

This module contains functions to interact directly with the VAPP-Virtual Application.

- ```
  (va::vaeng::run-string str)
  (va::parse-multipart-body pbuf body)
  dispatcher (mtfa-dispatcher::make eis::BaseLib::BlockSession
  (define va::vaeng::AddToSet #nil)
  ```
  ```

```
(define va::vaeng::AddToDict #nil)
(define va::vaeng::DelFromSet #nil)
(define va::vaeng::GetFromSet #nil)
(define va::vaeng::SendRawData #nil)
```
These functions are analogous to the ARS functions. So you can modify a set of strings in the ARS module, directly from this module.

3.9 module extset, umset

This module contains class and methods to interact with a special extended set data structure.

- To make it: (make <mtfa-extset>)

```
(mtfa-extset::clone-empty (ds <mtfa-extset>))
(mtfa-extset::set (ds <mtfa-extset>) (skey <string>) (setunset <boolean>))
(mtfa-extset::set (ds <mtfa-extset>) (lskey <list>) (setunset <boolean>))
(mtfa-extset::check (ds <mtfa-extset>) (skey <string>))
(mtfa-extset::get-all (ds <mtfa-extset>))
(mtfa-extset::get-universal-all (ds <mtfa-extset>))
(mtfa-extset::and (ds <mtfa-extset>) (ds1 <mtfa-extset>))
(mtfa-extset::or (ds <mtfa-extset>) (ds1 <mtfa-extset>))
(mtfa-extset::xor (ds <mtfa-extset>) (ds1 <mtfa-extset>))
(mtfa-extset::not (ds <mtfa-extset>))
(mtfa-extset::get-set (ds <mtfa-extset>))
(mtfa-extset? (ds <mtfa-extset>))
(mtfa-extset::serialize-locals (ds <mtfa-extset>))
(mtfa-extset::serialize-globals (ds <mtfa-extset>))
(mtfa-extset-make-instance cname)

Usage:
;;makes an extset labeled "extset-test". If you makes more extset using the same label, they
share the same universal set
(define es (mtfa-extset-make-instance "extset-test"))
(mtfa-extset::set es "Hello" #t) ;;or set to #f
(mtfa-extset::set es "Green" #t) ;;or set to #f
(mtfa-extset::set es "Garden" #t) ;;or set to #f
(mtfa-extset::set es "Sky" #t) ;;or set to #f

(mtfa-extset::get-all es)
$17 = ("Hello" "Green" "Garden" "Sky")
```

3.10 module web

- the reference class is <mtfa-web-security-base-blum>

- The methods are:

```
(mwsb::make-residue (ds <mtfa-web-security-base-blum>) (num <number>))
(mwsb::next (ds <mtfa-web-security-base-blum>) (num <number>))
(mwsb::next-ith (ds <mtfa-web-security-base-blum>) (num <number>) (idx <number> ))
(mwsb::prev (ds <mtfa-web-security-base-blum>) (num <number>))
(mwsb::prev-ith (ds <mtfa-web-security-base-blum>) (num <number>) (idx <number> ))
(mwsb::make-token (ds <mtfa-web-security-base-blum>) (base <string>) (timeout <number> ))
(mwsb::identify-token (ds <mtfa-web-security-base-blum>) token)
(mwsb::cypher (ds <mtfa-web-security-base-blum>) (bv <bytevector>))
(mwsb::decypher (ds <mtfa-web-security-base-blum>) (bv <bytevector>) (code <number>))
```

- A more general class is defined: <mtfa-web-cookie-maker> whose methods are:

```
(mwcm::make-a-cookie (ds <mtfa-web-cookie-maker>)
                                          (key <string>)
                                          (expiration <string>)
                                          (duration <number>)
                                          (path <string>)
                                          (domain <string>)
                                          (http_only <boolean>)
                                          (secure <boolean>)
                                          )
        (mwcm::check-a-cookie (ds <mtfa-web-cookie-maker>) (value <string>))
```

- A JSON responder: (mtfa-web-make-json-requests-manager va-key #:key (format-error GENERAL-JSON-ANS
  (auxiliary-info '()))
  (GENERAL-JSON-ANSWER::error msg)
  (GENERAL-JSON-ANSWER::success msg data_name data)

- To manage WEB sessions:
  (mtfa-session-id-make duration string)
  (mtfa-session-id-check full-sid)

- To serialize HTML forms: (mtfa-sxml2html tree)
  Returns the serialized HTML form of TREE
  EXAMPLES
  (sxml2html `(div (@ (class \"p-2 hoverable border rounded\") (style \"height: auto;width:
  auto;\"))
    (div (@ (class \"mb-0\")) ,(string-append label \"(\" units \")\"))
    (input (@ (style \"width: 90%;\")(type \"text\") (id ,id) (name ,id) (class \"form-control\")
  (autocomplete \"nope\") (value \"\")))))
  (sxml2html `(div (@ (class \"p-2 range-field hoverable border rounded\") (style \"height:
  auto;width: auto;\"))
    (div (@ (class \"row mb-3\"))
         (div (@ (class \"mb-0 col-9\")) ,(string-append label \"(\" units \")\"))
         (fiv (@ (class \"mb-0 col-2\") (id ,(string-append \"slider-value-\" id))) \"\"))
    (input (@ (value \"-1\") (type \"range\") (min ,(vector-ref min-max 0)) (max ,(vector-ref
  min-max 1))
      (id ,id) (name ,id) (class \"a-slider form-control col-12\")))))
  (mtfa-sxml2html '((doctype \"html\")(html (head (raw \"<meta http-equiv=\\\"Content-Type\\\"
  content=\\\"text/html; charset=UTF-8\\\">\") (meta (@ (charset \"UTF-8\"))) (title \"Title\"))
  (body ((one 1)(two 2))(three (@ (a 10))3)))))

3.11 module nn

The neural network module

- Some functions:
  mtfa-msec->sdate (msecs)
  mtfa-usec->sdate (usecs)
  mtfa-nsec->sdate (nsecs)
  mtfa-usec->json (usecs)
  mtfa-gaussian-generator

- The main neural netowkr builder
  mtfa-nn (inodes hnodes onodes lr) => generates a closure that gives:
  'show
  'query l-inputs
  'train l-inputs l-target
  if starting with a:: the nn uses the GPU

3.12 module avl

The avl data structure

- (mtfa-avl-make less-proc)
  Makes an avl-tree with search, delete, insert, ... Wants the less function.
  usage: (define avl (mtfa-avl-make <))

  (avl-search avl key . less)
  (avl-insert avl key value)
  (avl-delete avl key)
  (avl-dump avl)
  (avl-print avl)
  (avl-map avl f)
  (avl-for-each avl f)
  (avl-clear avl)

  (avl-next node)
  (avl-prec node)
  (avl-value node)
  (avl-value! node value)
  (avl-key node)
  (avl-key! node key)
  (avl-kv node)
  (avl->list avl)
  (avl-clone avl)
  (avl-first avl)
  (avl-last avl)
  (avl-union avl1 avl2 . less)
  (avl-intersection avl1 avl2 . less)
  (avl-before avl k)
  (avl-after avl k)
  (avl-size avl)

3.13 module eqt

The quad tree data structure

- (mtfa-eqt-make lato #:optional (indexid #f))
  (mtfa-eqt-clone eqt)
  (mtfa-eqt-make-from-unitary-cells lato id-x-y #:optional (indexid #f))
  (mtfa-eqt-merge eqt1 eqt2)
  (mtfa-eqt-area eqt)
  (mtfa-eqt-make-from-rects lato listOfRects #:optional (indexid #t))
  (mtfa-eqt-list-intersections eqt1 eqt2)
  (mtfa-eqt-intersect? eqt1 eqt2)
  (mtfa-eqt-list-regions eqt id)
  (mtfa-eqt-give-rindex eqt)
  (mtfa-eqt-show name eqt)

3.14 module opencv

A module for computer vision and deep learning

- (define-public mtfa-opencv::interpolator-types::_2D::ThinPlateSplineInterpolator 0)
  (define-public mtfa-opencv::interpolator-types::_2D::BicubicInterpolator 1)
  (define-public mtfa-opencv::interpolator-types::_2D::BilinearInterpolator 2)
  (define-public mtfa-opencv::interpolator-types::_1D::CubicSplineInterpolator 3)
  (define-public mtfa-opencv::interpolator-types::_1D::LinearInterpolator 4)

```
(define-public mtfa-opencv::interpolator-types::_1D::MonotonicInterpolator 5)

(define-public (mtfa-opencv-make-interpolator type)
  "(mtfa-opencv-make-interpolator type) where types are _2D::ThinPlateSplineInterpolator,
_2D::BicubicInterpolator, _2D::BilinearInterpolator, _1D::CubicSplineInterpolator, _1D::LinearInte
_1D::MonotonicInterpolator"
  (mtfa_opencv_make_interpolator type))

(define-public (mtfa-opencv-interpolator-add-serie interpolator list-of-series)
  "(mtfa-opencv-interpolator-add-serie type list-of-series) where interpolator has to have
been created with mtfa-opencv-make-interpolator and the list is like ((x1 x2 ...)(y1 y2 ...)(z1
z2 ...)). Where zi is the value objective of the interpolation"
  (mtfa_opencv_interpolator_add_serie interpolator list-of-series))

(define-public (mtfa-opencv-interpolator-interpolate interpolator list-of-coordinates)
  "(mtfa-opencv-interpolator-interpolate type list-of-coordinates) where interpolator has
to have been created with mtfa-opencv-make-interpolator and the list is like (x y). the interpolat
value is returned"
  (mtfa_opencv_interpolator_interpolate interpolator list-of-coordinates))

;;Features e best matching
(define-public (mtfa-opencv-features-detector img)
  "Finds features using FAST detector and BEBLID descriptor algorithms. Returns two values:
descriptors and points"
  (mtfa_opencv_features_detector img))

(define* (mtfa-opencv-features-find-transformation descr1 points1 descr2 points2 #:key (norm
6) (method 8))
  "Finds the number of matches and the number of inliers obtained matching points from the
two set of features detected.
descr1/2 and points1/2 are obtained from the mtfa-opencv-features-detector.
Norm is an integer: NORM_INF 1, NORM_L1 2, NORM_L2 4, NORM_L2SQR 5, NORM_HAMMING 6, NORM_HAMMING2
7, NORM_RELATIVE 8, NORM_MINMAX 32 (default HAMMING).
Method is an integer (default RANSAC):
       DEFAULT 0,
       LMEDS  = 4,  least-median of squares algorithm
       RANSAC = 8,  RANSAC algorithm
       RHO    = 16, RHO algorithm
       USAC_DEFAULT = 32,  USAC algorithm, default settings
       USAC_PARALLEL = 33, USAC, parallel version
       USAC_FM_8PTS = 34,  USAC, fundamental matrix 8 points
       USAC_FAST = 35,     USAC, fast settings
       USAC_ACCURATE = 36, USAC, accurate settings
       USAC_PROSAC = 37,   USAC, sorted points, runs PROSAC
       USAC_MAGSAC = 38    USAC, runs MAGSAC++
"
  (mtfa_opencv_features_find_transformation descr1 points1 descr2 points2 norm method))
(export mtfa-opencv-features-find-transformation)

;;I ph-tree
(define-public (mtfa-phtree-box-2d-make)
  "Makes a ph-tree for 2d boxes"
  (let ((db (make-hash-table)))
    (cons db (mtfa_phtree_box_2d_make))))

(define-public (mtfa-phtree-box-3d-make)
  "Makes a ph-tree for 3d boxes"
  (let ((db (make-hash-table)))
    (cons db (mtfa_phtree_box_3d_make))))

(define-public (mtfa-phtree-point-2d-make)
```

```
  "Makes a ph-tree for 2d points"
  (let ((db (make-hash-table)))
    (cons db (mtfa_phtree_point_2d_make))))

(define-public (mtfa-phtree-point-3d-make)
  "Makes a ph-tree for 3d points"
  (let ((db (make-hash-table)))
    (cons db (mtfa_phtree_point_3d_make))))

(define-public (mtfa-phtree-2d-add-points ph-tree lop)
  "adds 2d points list to the tree: '((x0 y0 id0)(x1 y1 id1)...)"
  (for-each (lambda (pt)
      (hash-set! (car ph-tree) (last pt) pt))
    lop)
  (mtfa_phtree_2d_add_points (cdr ph-tree) lop))

(define-public (mtfa-phtree-3d-add-points ph-tree lop)
  "adds 3d points list to the tree: '((x0 y0 z0 id0)(x1 y1 z1 id1)...)"
  (for-each (lambda (pt)
      (hash-set! (car ph-tree) (last pt) pt))
    lop)
  (mtfa_phtree_3d_add_points (cdr ph-tree) lop))

(define-public (mtfa-phtree-2d-add-boxes ph-tree lob)
  "adds 2d boxes list to the tree: '(((xmin ymin)(xmax ymax) id) ((xmin ymin)(xmax ymax)
id) ...)"
  (for-each (lambda (pt)
      (hash-set! (car ph-tree) (last pt) pt))
    lob)
  (mtfa_phtree_2d_add_boxes (cdr ph-tree) lob))

(define-public (mtfa-phtree-3d-add-boxes ph-tree lob)
  "adds 3d boxes list to the tree: '(((xmin ymin zmin)(xmax ymax zmax) id) ((xmin ymin zmin)(xmax
ymax zmax) id) ...)"
  (for-each (lambda (pt)
      (hash-set! (car ph-tree) (last pt) pt))
    lob)
  (mtfa_phtree_3d_add_boxes (cdr ph-tree) lob))

(define-public (mtfa-phtree-2d-query-points ph-tree min-range max-range)
  "Queries 2d points contained in the box delimited by min-range (x y) and max-range (x y)"
  (mtfa_phtree_2d_query_points (cdr ph-tree) min-range max-range))

(define-public (mtfa-phtree-3d-query-points ph-tree min-range max-range)
  "Queries 3d points contained in the box delimited by min-range (x y z) and max-range (x
y z)"
  (mtfa_phtree_3d_query_points (cdr ph-tree) min-range max-range))

(define-public (mtfa-phtree-2d-query-boxes ph-tree min-range max-range included)
  "Queries 2d boxes contained in the box delimited by min-range (x y) and max-range (x y).
Included is booleand and defines if completely included or if intersected"
  (mtfa_phtree_2d_query_boxes (cdr ph-tree) min-range max-range included))

(define-public (mtfa-phtree-3d-query-boxes ph-tree min-range max-range included)
  "Queries 3d boxes contained in the box delimited by min-range (x y z) and max-range (x
y z). Included is booleand and defines if completely included or if intersected"
  (mtfa_phtree_3d_query_boxes (cdr ph-tree) min-range max-range included))

(define-public (mtfa-phtree-2d-knn-points ph-tree min-result-size center)
  "gives at least min-result-size points as list of (id . distance) ordered by min distance
from the point (x y) given as center."
```

```
       (mtfa_phtree_2d_knn_points (cdr ph-tree) min-result-size center))

(define-public (mtfa-phtree-3d-knn-points ph-tree min-result-size center)
  "gives at least min-result-size points as list of (id . distance) ordered by min distance
from the point (x y z) given as center."
       (mtfa_phtree_3d_knn_points (cdr ph-tree) min-result-size center))

(define-public (mtfa-phtree-2d-getall-points ph-tree)
  "Gives the ids of all the 2d points"
  (hash-map->list (lambda (k v) k) (car ph-tree)))

(define-public (mtfa-phtree-3d-getall-points ph-tree)
  "Gives the ids of all the 3d points"
  (hash-map->list (lambda (k v) k) (car ph-tree)))

(define-public (mtfa-phtree-2d-getall-boxes ph-tree)
  "Gives the ids of all the 2d boxes"
  (hash-map->list (lambda (k v) k) (car ph-tree)))

(define-public (mtfa-phtree-3d-getall-boxes ph-tree)
  "Gives the ids of all the 3d boxes"
  (hash-map->list (lambda (k v) k) (car ph-tree)))

(define-public (mtfa-phtree-2d-get-point ph-tree pid)
  "Gives the complete data associated to the id"
  (hash-ref (car ph-tree) pid))

(define-public (mtfa-phtree-3d-get-point ph-tree pid)
  "Gives the complete data associated to the id"
  (hash-ref (car ph-tree) pid))

(define-public (mtfa-phtree-2d-get-box ph-tree pid)
  "Gives the complete data associated to the id"
  (hash-ref (car ph-tree) pid))

(define-public (mtfa-phtree-3d-get-box ph-tree pid)
  "Gives the complete data associated to the id"
  (hash-ref (car ph-tree) pid))

(define-public (mtfa-phtree-2d-erase-points ph-tree list-of-pid)
  (mtfa_phtree_2d_erase_points (cdr ph-tree)
       (remove not (map (lambda (it) (hash-ref (car ph-tree) it)) list-of-pid)))
  (for-each (lambda (i) (hash-remove! (car ph-tree) i)) list-of-pid))
(define-public (mtfa-phtree-3d-erase-points ph-tree list-of-pid)
  (mtfa_phtree_3d_erase_points (cdr ph-tree)
       (remove not (map (lambda (it) (hash-ref (car ph-tree) it)) list-of-pid)))
  (for-each (lambda (i) (hash-remove! (car ph-tree) i)) list-of-pid))
(define-public (mtfa-phtree-2d-erase-boxes ph-tree list-of-pid)
  (mtfa_phtree_2d_erase_boxes (cdr ph-tree)
       (remove not (map (lambda (it) (hash-ref (car ph-tree) it)) list-of-pid)))
  (for-each (lambda (i) (hash-remove! (car ph-tree) i)) list-of-pid))
(define-public (mtfa-phtree-3d-erase-boxes ph-tree list-of-pid)
  (mtfa_phtree_3d_erase_boxes (cdr ph-tree)
      (remove not (map (lambda (it) (hash-ref (car ph-tree) it)) list-of-pid)))
  (for-each (lambda (i) (hash-remove! (car ph-tree) i)) list-of-pid))

(define (mtfa-opencv-detector-p detector) (mtfa_opencv_detector_p detector))
(define (mtfa-opencv-detector-make cfg weights) (mtfa_opencv_detector_make cfg weights))
(define (mtfa-opencv-detector-run detector par-list) (mtfa_opencv_detector_run detector par-list))
(define (mtfa-opencv-detector-stop detector) (mtfa_opencv_detector_stop detector))
(define (mtfa-opencv-detector-get-zed-camera detector cameranum) (mtfa_opencv_detector_get_zed_cam
```

```
detector cameranum))

(define (mtfa-zed2-detector-p detector) (mtfa_zed2_detector_p detector))
(define (mtfa-zed2-detector-make) (mtfa_zed2_detector_make))
(define (mtfa-zed2-detector-run detector par-list) (mtfa_zed2_detector_run detector par-list))
(define (mtfa-zed2-detector-stop detector) (mtfa_zed2_detector_stop detector))
(define (mtfa-zed2-detector-get-zed-camera detector cameranum) (mtfa_zed2_detector_get_zed_camera
detector cameranum))

(define (mtfa-yolo4-detector-p detector) (mtfa_yolo4_detector_p detector))
(define (mtfa-yolo4-detector-make) (mtfa_yolo4_detector_make))
(define (mtfa-yolo4-detector-run detector par-list) (mtfa_yolo4_detector_run detector par-list))
(define (mtfa-yolo4-detector-stop detector) (mtfa_yolo4_detector_stop detector))

(define (mtfa-opencv-imgfile-to-mat imgpath) (mtfa_opencv_imgfile_to_mat imgpath))
(define (mtfa-opencv-mat-to-imgfile mat filepath) (mtfa_opencv_mat_to_imgfile mat filepath))
(define (mtfa-opencv-crop-rectangle mat tlx tly width height) (mtfa_opencv_crop_rectangle
mat tlx tly width height))
;;(define (mtfa-opencv-get-raw-cloud mat) (mtfa_opencv_get_raw_cloud mat))
(define (mtfa-opencv-binary-op m1 m2 op) (mtfa_opencv_binary_op m1 m2 op))
(define-public mtfa-opencv::binary-op::ADD 0)
(define-public mtfa-opencv::binary-op::SUB 1)
(define-public mtfa-opencv::binary-op::AND 2)
(define-public mtfa-opencv::binary-op::OR 3)
(define-public mtfa-opencv::binary-op::XOR 4)
(define-public mtfa-opencv::binary-op::MUL 5)
(define-public mtfa-opencv::binary-op::DIV 6)
(define (mtfa-opencv-compute-image-hash-AverageHash mat) (mtfa_opencv_compute_image_hash_AverageHa
mat))
(define (mtfa-opencv-compare-image-hash-AverageHash hash-1 hash-2) (mtfa_opencv_compare_image_hash
hash-1 hash-2))
(define (mtfa-opencv-compute-image-hash-BlockMeanHash0 mat) (mtfa_opencv_compute_image_hash_BlockM
mat))
(define (mtfa-opencv-compare-image-hash-BlockMeanHash0 hash-1 hash2) (mtfa_opencv_compare_image_ha
hash-1 hash2))
(define (mtfa-opencv-compute-image-hash-BlockMeanHash1 mat) (mtfa_opencv_compute_image_hash_BlockM
mat))
(define (mtfa-opencv-compare-image-hash-BlockMeanHash1 hash-1 hash2) (mtfa_opencv_compare_image_ha
hash-1 hash2))
(define (mtfa-opencv-compute-image-hash-ColorMomentHash mat) (mtfa_opencv_compute_image_hash_Color
mat))
(define (mtfa-opencv-compare-image-hash-ColorMomentHash hash-1 hash2) (mtfa_opencv_compare_image_h
hash-1 hash2))
(define (mtfa-opencv-compute-image-hash-MarrHildrethHash mat) (mtfa_opencv_compute_image_hash_Marr
mat))
(define (mtfa-opencv-compare-image-hash-MarrHildrethHash hash-1 hash2) (mtfa_opencv_compare_image_
hash-1 hash2))
(define (mtfa-opencv-compute-image-hash-PHash mat) (mtfa_opencv_compute_image_hash_PHash
mat))
(define (mtfa-opencv-compare-image-hash-PHash hash-1 hash2) (mtfa_opencv_compare_image_hash_PHash
hash-1 hash2))
(define (mtfa-opencv-compute-image-hash-RadialVarianceHash mat) (mtfa_opencv_compute_image_hash_Ra
mat))
(define (mtfa-opencv-compare-image-hash-RadialVarianceHash hash-1 hash2) (mtfa_opencv_compare_imag
hash-1 hash2))
(define (mtfa-opencv-img-hash-to-bytevector mat) (mtfa_opencv_img_hash_to_bytevector mat))
(define (mtfa-opencv-bytevector-to-img-hash bv) (mtfa_opencv_bytevector_to_img_hash bv))
(define (mtfa-opencv-make-mat-zeros rows cols bytelen type colors) (mtfa_opencv_make_mat_zeros
rows cols bytelen type colors))
(define (mtfa-opencv-make-empty-mat) (mtfa_opencv_make_empty_mat))
(define (mtfa-opencv-draw-imshow wname mat) (mtfa_opencv_draw_imshow wname mat))
```

```
(define (mtfa-opencv-draw-move-window wname x y) (mtfa_opencv_draw_move_window wname x y))
(define (mtfa-opencv-draw-wait-key ms) (mtfa_opencv_draw_wait_key ms))
(define (mtfa-opencv-draw-rectangle mat x y w h rgb thick linetype) (mtfa_opencv_draw_rectangle
mat x y w h rgb thick linetype))
(define (mtfa-opencv-draw-polygon mat points ncontours isClosed rgb thick linetype)  (mtfa_opencv_
mat points ncontours isClosed rgb thick linetype))
(define (mtfa-opencv-fill-polygons mat list-of-poly rgb)  (mtfa_opencv_fill_polygons mat
list-of-poly rgb))
(define (mtfa-opencv-fill-convex-polygon mat points rgb)  (mtfa_opencv_fill_convex_polygon
mat points rgb))
(define (mtfa-opencv-draw-circle mat x y radius rgb thick linetype) (mtfa_opencv_draw_circle
mat x y radius rgb thick linetype))
(define (mtfa-opencv-draw-ellipse mat x y axes angle start_angle end_angle rgb thick linetype)
(mtfa_opencv_draw_ellipse mat x y axes angle start_angle end_angle rgb thick linetype))
(define (mtfa-opencv-draw-destroy-window name) (mtfa_opencv_draw_destroy_window name))
(define (mtfa-opencv-draw-destroy-all-windows) (mtfa_opencv_draw_destroy_all_windows))
(define (mtfa-opencv-draw-named-window name type) (mtfa_opencv_draw_named_window name type))
(define (mtfa-opencv-draw-resize-window name w h) (mtfa_opencv_draw_resize_window name w
h))
(define (mtfa-opencv-draw-put-text mat text x y fontFace fontScale rgb thick linetype) (mtfa_opencv
mat text x y fontFace fontScale rgb thick linetype))
(define (mtfa-opencv-draw-line mat x1 y1 x2 y2 rgb thick linetype) (mtfa_opencv_draw_line
mat x1 y1 x2 y2 rgb thick linetype))
(define (mtfa-opencv-mat-info mat) (mtfa_opencv_mat_info mat))
(define (mtfa-opencv-cap-info cap)
  "returns list of
POS_MSEC Current position of the video file in milliseconds.
POS_FRAMES 0-based index of the frame to be decoded/captured next.
POS_AVI_RATIO Relative position of the video file: 0=start of the film, 1=end of the film.
FRAME_WIDTH Width of the frames in the video stream.
FRAME_HEIGHT Height of the frames in the video stream.
FPS Frame rate.
FOURCC 4-character code of codec. see VideoWriter::fourcc .
FRAME_COUNT Number of frames in the video file."
  (mtfa_opencv_cap_info cap))
(define (mtfa-opencv-draw-display-overlay winname text ms) (mtfa_opencv_draw_display_overlay
winname text ms))
(define (mtfa-opencv-draw-display-status-bar winname text ms) (mtfa_opencv_draw_display_status_bar
winname text ms))
(define (mtfa-opencv-draw-set-window-property winname property value) (mtfa_opencv_draw_set_window
winname property value))
(define* (mtfa-lap table #:optional (infinite -1)) (mtfa_lap table infinite))



(define (mtfa-opencv-change-brightness mat added) (mtfa_opencv_change_brightness mat added))
(define (mtfa-opencv-change-contrast mat multiplied) (mtfa_opencv_change_contrast mat multiplied))
(define (mtfa-opencv-change-brightness-and-contrast mat brightness:0..200 contrast:0..200)
"Brightness 0..200, Contrast 0..200" (mtfa_opencv_change_brightness_and_contrast mat brightness:0.
contrast:0..200))
(define (mtfa-opencv-resize mat resize_x resize_y) (mtfa_opencv_resize mat resize_x resize_y))
(define (mtfa-opencv-prop-resize mat resize_x) (mtfa_opencv_prop_resize mat resize_x))
(define (mtfa-opencv-gray mat) (mtfa_opencv_gray mat ))
(define (mtfa-opencv-add-gaussian-noise mat sigma mean) (mtfa_opencv_add_gaussian_noise mat
sigma mean))
(define (mtfa-opencv-salt-and-pepper mat pa pb) (mtfa_opencv_salt_and_pepper mat pa pb))
(define (mtfa-opencv-gaussian-blur mat sigma_x sigma_y) (mtfa_opencv_gaussian_blur mat sigma_x
sigma_y))
(define (mtfa-opencv-median-blur mat ksize) (mtfa_opencv_median_blur mat ksize))
(define (mtfa-opencv-rotate mat angle ) (mtfa_opencv_rotate mat angle ))
(define (mtfa-opencv-erode mat x_size y_size iterations) (mtfa_opencv_erode mat x_size y_size
```

```
iterations))
(define (mtfa-opencv-contours mat canny blur-size-3) (mtfa_opencv_contours mat canny blur-size-3))
(define (mtfa-opencv-dilate mat x_size y_size iterations) (mtfa_opencv_dilate mat x_size
y_size iterations))
(define (mtfa-opencv-transform mat tl_x tl_y tr_x tr_y br_x br_y bl_x bl_y) (mtfa_opencv_transform
mat tl_x tl_y tr_x tr_y br_x br_y bl_x bl_y))
(define (mtfa-opencv-gamma-correction mat gamma ) (mtfa_opencv_gamma_correction mat gamma
))
(define (mtfa-opencv-equalize-histogram mat) (mtfa_opencv_equalize_histogram mat))
(define (mtfa-opencv-add-watermark img wat center_x_rel center_y_rel width_rel height_rel)
(mtfa_opencv_add_watermark img wat center_x_rel center_y_rel width_rel height_rel))
(define (mtfa-opencv-jpeg-compression mat quality ) (mtfa_opencv_jpeg_compression mat quality
))

(define* (mtfa-opencv-open-input-stream stream #:optional (preferencies '())) (if (not (null?
preferencies))
  (mtfa_opencv_open_input_stream stream preferencies)
  (mtfa_opencv_open_input_stream stream)))
(define (mtfa-opencv-close-input-stream opened-stream) (mtfa_opencv_close_input_stream opened-strea
(define (mtfa-opencv-read-input-stream mat opened-stream)(mtfa_opencv_read_input_stream mat
opened-stream))
(define (mtfa-opencv-input-stream-is-opened opened-stream) (mtfa_opencv_input_stream_is_opened
opened-stream))

(define (mtfa-opencv-open-output-stream stream fourcc fps size is_color) (mtfa_opencv_open_output_s
stream fourcc fps size is_color))
(define (mtfa-opencv-close-output-stream opened-stream) (mtfa_opencv_close_output_stream
opened-stream))
(define (mtfa-opencv-write-output-stream mat opened-stream) (mtfa_opencv_write_output_stream
mat opened-stream))
(define (mtfa-opencv-output-stream-is-opened opened-stream) (mtfa_opencv_output_stream_is_opened
opened-stream))

(define (mtfa-opencv-make-polygon list-of-xypairs) (mtfa_opencv_make_polygon list-of-xypairs))
(define (mtfa-opencv-point-in-polygon xypair polygon) (mtfa_opencv_point_in_polygon xypair
polygon))
(define (mtfa-opencv-compose im1 im2 vertical) (mtfa_opencv_compose im1 im2 vertical))
(define (mtfa-opencv-clone im1) (mtfa_opencv_clone im1))

(define (mtfa-opencv-rgbtohsv int_R int_G int_B) (mtfa_opencv_rgbtohsv int_R int_G int_B))
(define (mtfa-opencv-create-trackbar string_label string_window_name int_initial_value int_max_valu
call_TrackbarCallback void_userdata) (mtfa_opencv_create_trackbar string_label string_window_name
int_initial_value int_max_value call_TrackbarCallback void_userdata))

(define (mtfa-opencv-set-trackbar-max string_label string_window_name int_value) (mtfa_opencv_set_t
string_label string_window_name int_value))
(define (mtfa-opencv-set-trackbar-min string_label string_window_name int_value) (mtfa_opencv_set_t
string_label string_window_name int_value))
(define (mtfa-opencv-set-trackbar-pos string_label string_window_name int_value) (mtfa_opencv_set_t
string_label string_window_name int_value))
(define (mtfa-opencv-get-trackbar-pos string_label string_window_name) (mtfa_opencv_get_trackbar_po
string_label string_window_name))

(define*
  (mtfa-opencv-find-circles mat #:optional (int_dp:2 '()) (int_minDist:54 '()) (int_p1:90
'()) (int_p2:40 '()) (int_minr:7 '()) (int_maxr:19 '()) (int_blur:5 '()) (polygon '()))
  (mtfa_opencv_find_circles mat int_dp:2 int_minDist:54 int_p1:90 int_p2:40 int_minr:7 int_maxr:19
int_blur:5 polygon))

(define (mtfa-opencv-img-mean mat_img int_center_x int_center_y int_side) (mtfa_opencv_img_mean
mat_img int_center_x int_center_y int_side))
```

```
(define (mtfa-opencv-set-mouse-callback string_window_name call_MouseCallback void_userdata)
(mtfa_opencv_set_mouse_callback string_window_name call_MouseCallback void_userdata))

;;I parametri utilizzati dalle funzioni OPENCV
(define-public mtfa-opencv::window-flags::WINDOW_NORMAL #x00000000)
(define-public mtfa-opencv::window-flags::WINDOW_AUTOSIZE #x00000001)
(define-public mtfa-opencv::window-flags::WINDOW_OPENGL #x00001000)
(define-public mtfa-opencv::window-flags::WINDOW_FULLSCREEN 1)
(define-public mtfa-opencv::window-flags::WINDOW_FREERATIO #x00000100)
(define-public mtfa-opencv::window-flags::WINDOW_KEEPRATIO #x00000000)
(define-public mtfa-opencv::window-flags::WINDOW_GUI_EXPANDED #x00000000)
(define-public mtfa-opencv::window-flags::WINDOW_GUI_NORMAL #x00000010)

(define-public mtfa-opencv::window-property-flags::WND_PROP_FULLSCREEN 0)
(define-public mtfa-opencv::window-property-flags::WND_PROP_AUTOSIZE 1)
(define-public mtfa-opencv::window-property-flags::WND_PROP_ASPECT_RATIO 2)
(define-public mtfa-opencv::window-property-flags::WND_PROP_OPENGL 3)
(define-public mtfa-opencv::window-property-flags::WND_PROP_VISIBLE 4)
(define-public mtfa-opencv::window-property-flags::WND_PROP_TOPMOST 5)

(define-public mtfa-opencv::MOUSE_EVENT_FLAG_LBUTTON 1)
(define-public mtfa-opencv::MOUSE_EVENT_FLAG_RBUTTON 2)
(define-public mtfa-opencv::MOUSE_EVENT_FLAG_MBUTTON 4)
(define-public mtfa-opencv::MOUSE_EVENT_FLAG_CTRLKEY 8)
(define-public mtfa-opencv::MOUSE_EVENT_FLAG_SHIFTKEY 16)
(define-public mtfa-opencv::MOUSE_EVENT_FLAG_ALTKEY 32)

(define-public mtfa-opencv::EVENT_MOUSEMOVE 0)
(define-public mtfa-opencv::EVENT_LBUTTONDOWN 1)
(define-public mtfa-opencv::EVENT_RBUTTONDOWN 2)
(define-public mtfa-opencv::EVENT_MBUTTONDOWN 3)
(define-public mtfa-opencv::EVENT_LBUTTONUP 4)
(define-public mtfa-opencv::EVENT_RBUTTONUP 5)
(define-public mtfa-opencv::EVENT_MBUTTONUP 6)
(define-public mtfa-opencv::EVENT_LBUTTONDBLCLK 7)
(define-public mtfa-opencv::EVENT_RBUTTONDBLCLK 8)
(define-public mtfa-opencv::EVENT_MBUTTONDBLCLK 9)
(define-public mtfa-opencv::EVENT_MOUSEWHEEL 10)
(define-public mtfa-opencv::EVENT_MOUSEHWHEEL 11)

(define-public mtfa-opencv::QT_PUSH_BUTTON 0)
(define-public mtfa-opencv::QT_CHECKBOX 1)
(define-public mtfa-opencv::QT_RADIOBOX 2)
(define-public mtfa-opencv::QT_NEW_BUTTONBAR 1024)

(define-public mtfa-opencv::QT_STYLE_NORMAL 0)
(define-public mtfa-opencv::QT_STYLE_ITALIC 1)
(define-public mtfa-opencv::QT_STYLE_OBLIQUE 2)


(define-public mtfa-opencv::QT_FONT_LIGHT 25)
(define-public mtfa-opencv::QT_FONT_NORMAL 50)
(define-public mtfa-opencv::QT_FONT_DEMIBOLD 63)
(define-public mtfa-opencv::QT_FONT_BOLD 75)
(define-public mtfa-opencv::QT_FONT_BLACK 87)

;;La ZED CAMERA
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::BRIGHTNESS 0)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::CONTRAST 1)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::HUE 2)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::SATURATION 3)
```

```
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::SHARPNESS 4)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::GAMMA 5)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::GAIN 6)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::EXPOSURE 7)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::AEC_AGC 8)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::AEC_AGC_ROI 9)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::WHITEBALANCE_TEMPERATURE 10)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::WHITEBALANCE_AUTO 11)
(define-public mtfa-opencv-zed::VIDEO_SETTINGS::LED_STATUS 12)


(define-public (mtfa-opencv-zed-camera-grab zed-camera-pointer) (mtfa_opencv_zed_camera_grab
 zed-camera-pointer))
(define-public (mtfa-opencv-zed-camera-get-image zed-camera-pointer) (mtfa_opencv_zed_camera_get_i
zed-camera-pointer))

(define-public (mtfa-opencv-zed-camera-set zed-camera-pointer video-setting value) (mtfa_opencv_ze
zed-camera-pointer video-setting value))
(define-public (mtfa-opencv-zed-camera-get zed-camera-pointer video-setting) (mtfa_opencv_zed_came
zed-camera-pointer video-setting))
(define-public (mtfa-opencv-zed-camera-reset zed-camera-pointer) (mtfa_opencv_zed_camera_reset
zed-camera-pointer))
(define-public (mtfa-zed2-detector-get-zed-camera zed2-detector camera-num) (mtfa_zed2_detector_ge
zed2-detector camera-num))
(define-public (mtfa-opencv-detector-get-zed-camera detector camera-num) (mtfa_opencv_detector_get
detector camera-num))

(define-public (mtfa-opencv-levenshtein-distance str1 str2)
"Calcola il numero minimo di insert/replace/delete necessarie a trasformare str1 in str2"
(mtfa_opencv_levenshtein_distance str1 str2))

(define* (mtfa-opencv-make-gpu-mat #:optional (rows '()) (cols '()) (fill '()))
(mtfa_opencv_make_gpu_mat rows cols fill))
(export mtfa-opencv-make-gpu-mat)


(define-public (mtfa-opencv-convert-mat-to-gpumat mat)
       (mtfa_opencv_convert_mat_to_gpumat mat) )
(define-public (mtfa-opencv-convert-gpumat-to-mat gpumat)
       (mtfa_opencv_convert_gpumat_to_mat gpumat))
(define-public (mtfa-opencv-make-gpumat-from-lolod listoflistofdouble)
       (mtfa_opencv_make_gpumat_from_lolod listoflistofdouble))
(define-public (mtfa-opencv-convert-gpumat-to-lolod gpumat) "convert from gpumat to list
of list of double"
       (mtfa_opencv_convert_gpumat_to_lolod gpumat))

(define-public (mtfa-opencv-gpumat-gemm src1 src1_t src2 src2_t alpha src3 src3_t beta)
       "src1*src2*alpha+src3*beta, gli elementi booleani _t indicano che la matrice va trasposta.
alpha e beta sono float. Se src2 è '(), non viene moltiplicato nulla a src1, tranne il valore
alpha. Se src3 è '(), non viene sommato nulla al risultato, tranne il valore beta"
(mtfa_opencv_gpumat_gemm src1 src1_t src2 src2_t alpha src3 src3_t beta))
(define-public (mtfa-opencv-transpose-gpumat src)
  (mtfa_opencv_transpose_gpumat src))
(define-public (mtfa-opencv-gpumat-rows-cols src)
  (mtfa_opencv_gpumat_rows_cols src))
(define-public (mtfa-opencv-gpumat-clone src)
  (mtfa_opencv_gpumat_clone src))
(define-public (mtfa-opencv-gpumat-math-add dst add1 add2)
  (mtfa_opencv_gpumat_math_add dst add1 add2))
(define-public (mtfa-opencv-gpumat-math-sub dst sub1 sub2)
  (mtfa_opencv_gpumat_math_sub dst sub1 sub2))
```

```
(define-public (mtfa-opencv-gpumat-math-mul dst mol1 mol2 scale)
  (mtfa_opencv_gpumat_math_mul dst mol1 mol2 scale))
(define-public (mtfa-opencv-gpumat-math-div dst div1 div2 scale)
  (mtfa_opencv_gpumat_math_div dst div1 div2 scale))
(define-public (mtfa-opencv-gpumat-math-absdiff dst sub1 sub2)
  (mtfa_opencv_gpumat_math_absdiff dst sub1 sub2))
(define-public (mtfa-opencv-gpumat-math-abs dst src)
  (mtfa_opencv_gpumat_math_abs dst src))
(define-public (mtfa-opencv-gpumat-math-sqr dst src)
  (mtfa_opencv_gpumat_math_sqr dst src))
(define-public (mtfa-opencv-gpumat-math-sqrt dst src)
  (mtfa_opencv_gpumat_math_sqrt dst src))
(define-public (mtfa-opencv-gpumat-math-exp dst src)
  (mtfa_opencv_gpumat_math_exp dst src))
(define-public (mtfa-opencv-gpumat-math-log dst src)
  (mtfa_opencv_gpumat_math_log dst src))
(define-public (mtfa-opencv-gpumat-math-pow dst src exp)
  (mtfa_opencv_gpumat_math_pow dst src exp))
(define-public (mtfa-opencv-gpumat-math-min dst op1 op2)
  (mtfa_opencv_gpumat_math_min dst op1 op2))
(define-public (mtfa-opencv-gpumat-math-max dst op1 op2)
  (mtfa_opencv_gpumat_math_max dst op1 op2))
(define-public (mtfa-opencv-gpumat-math-add-weighted dst add1 alpha add2 beta gamma)
  (mtfa_opencv_gpumat_math_add_weighted dst add1 alpha add2 beta gamma))

(define-public (mtfa-opencv-gpumat-make-diag side value)
   (mtfa_opencv_gpumat_make_diag side value))

(define-public mtfa-opencv::mat-info::CV_8U   0)
(define-public mtfa-opencv::mat-info::CV_8S   1)
(define-public mtfa-opencv::mat-info::CV_16U  2)
(define-public mtfa-opencv::mat-info::CV_16S  3)
(define-public mtfa-opencv::mat-info::CV_32S  4)
(define-public mtfa-opencv::mat-info::CV_32F  5)
(define-public mtfa-opencv::mat-info::CV_64F  6)
(define-public mtfa-opencv::mat-info::CV_15F  7)
```

- n-ary tree

```
(define-public (ntree-new data)
  (ntree-make data '() '() '() '()))

(export ntree-data
ntree-data!
ntree-next
ntree-next!
ntree-prev
ntree-prev!
ntree-parent
ntree-parent!
ntree-children
ntree-children! )
;;
;;
(define-public (ntree-copy node)
  ;;Recursively copies a GNode (but does not deep-copy the data inside the nodes, see g_node_copy_
if you need that).
  #f
  )
;;
(define-public (ntree-copy-deep node func)
```

```
  ;;Recursively copies a GNode and the data inside the nodes.
  #f
  )
;;
(define-public (ntree-insert parent position node)
  (ntree-parent! node parent)
  ;;(ntree-children! node '())
  (let ((pchi (ntree-children parent)))
    (cond
     ;;
     ((or (< position 0) (>= position (length pchi)) (zero? (length pchi)))
      (ntree-children! parent (append pchi (list node)))
      (ntree-next! node '())
      (when (> (length pchi) 0)
(ntree-prev! node (last pchi))
(ntree-next! (last pchi) node)))
     ;;
     ((= position 0)
      (ntree-next! node (first pchi))
      (ntree-prev! node '())
      (ntree-prev! (first pchi) node)
      (ntree-children! parent (append (list node) pchi)))
     ;;
     (#t
      (let ((prec (list-ref pchi (1- position)))
    (next (list-ref pchi position)))
(ntree-next! node next)
(ntree-prev! node prec)
(ntree-next! prec node)
(ntree-prev! next node)
(ntree-children! parent (append (take pchi position) (list node) (take-right pchi (- (length
pchi) position)))))))))

(define-public (ntree-insert-before parent sibling node)
  ;;If sibling is NULL, the node is inserted as the last child of parent
  (cond
   ((null? sibling)
    (ntree-insert parent -1 node))
   (#t (ntree-insert parent (list-index (cut eq? sibling <>) (ntree-children parent)) node))))
;;
(define-public (ntree-insert-after parent sibling node)
  ;;If sibling is NULL, the node is inserted as the last child of parent
  (cond
   ((null? sibling)
    (ntree-insert parent -1 node))
   (#t (ntree-insert parent (1+ (list-index (cut eq? sibling <>) (ntree-children parent)))
node))))
;;
(define-public (ntree-append parent node)
  (ntree-insert parent -1 node))
;;
(define-public (ntree-prepend parent node)
  (ntree-insert parent 0 node))
;;
(define-public (ntree-insert-data parent position data)
  (let ((node (ntree-new data)))
    (ntree-insert parent position node)
    node))
;;
(define-public (ntree-insert-data-before parent sibling data)
  (let ((node (ntree-new data)))
```

```
    (ntree-insert-before parent sibling node)
    node))
;;
(define-public (ntree-insert-data-after parent sibling data)
  (let ((node (ntree-new data)))
    (ntree-insert-after parent sibling node)
    node))
;;
(define-public (ntree-append-data parent data)
  (let ((node (ntree-new data)))
    (ntree-insert parent -1 node)
    node))
;;
(define-public (ntree-prepend-data parent data)
  (let ((node (ntree-new data)))
    (ntree-insert parent 0 node)
    node))
;;
(define-public (ntree-reverse-children node)
  (ntree-children! node
   (map (lambda (it) (let ((old-prev (ntree-prev it)) (old-next (ntree-next it)))
  (ntree-prev! it old-next)
  (ntree-next! it old-prev)
  ;; (Show! "Data: " (ntree-data it)
  ;;   ". Prev: " (if (null? (ntree-prev it)) "()" (ntree-data (ntree-prev it)))
  ;;   ". next: " (if (null? (ntree-next it)) "()" (ntree-data (ntree-next it)))
  ;;   )
  it))
     (reverse (ntree-children node)))))
(define-public ntree::TRAVERSE_LEAVES 'TRAVERSE_LEAVES)
(define-public ntree::TRAVERSE_ALL 'TRAVERSE_ALL)
(define-public ntree::TRAVERSE_NON_LEAVES 'TRAVERSE_NON_LEAVES)
(define-public ntree::IN_ORDER 'IN_ORDER)
(define-public ntree::PRE_ORDER 'PRE_ORDER)
(define-public ntree::POST_ORDER 'POST_ORDER)
(define-public ntree::LEVEL_ORDER 'LEVEL_ORDER)
;;
(defun-public ntree-traverse-or-find (root order flags max-depth func)
  (cond
   ((eqv? order 'IN_ORDER)
    (let loop ((r root) (depth 0))
      (if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
  '()
  (if (null? (ntree-children r))
      (if (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r))
  '())
      ;;visit all children except the last
      (begin
(for-each (lambda (it) (loop it (1+ depth)))
  (drop-right (ntree-children r) 1))
(when (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r))
  )
(loop (last (ntree-children r)) (1+ depth)))))))
   ;;
   ((eqv? order 'PRE_ORDER)
    (let loop ((r root) (depth 0))
      (if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
  '()
  (if (null? (ntree-children r))
```

```
      (if (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r))
  '())
      (begin
(when  (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r)))
;;visit all children
(for-each (lambda (it) (loop it (1+ depth)))
  (ntree-children r)))))))
   ;;
   ((eqv? order 'POST_ORDER)
    (let loop ((r root) (depth 0))
      (if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
  '()
  (if (null? (ntree-children r))
      (if (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r))
  '())
      (begin
(for-each (lambda (it) (loop it (1+ depth)))
  (ntree-children r))
(when  (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (when (func r depth) (return r)))))))
   ;;
   ((eqv? order 'LEVEL_ORDER)
    (if (null? root)
'()
(let loop ((nodes (list root))
   (depth 0))
  (if (null? nodes)
      '()
      ;;stampo la lista dei nodi attuali, faccio la lista dei figli, stampo la lista dei
figli ...
      (begin
(for-each (lambda (n)
    (cond
     ((eqv? flags 'TRAVERSE_ALL)
      (when (func n depth) (return n)))
     ((and (eqv? flags 'TRAVERSE_LEAVES) (null? (ntree-children n))) (when (func n depth)
(return n)))
     ((and (eqv? flags 'TRAVERSE_NON_LEAVES) (not (null? (ntree-children n)))) (when (func
n depth) (return n)))))
  nodes)
(loop (append-map (lambda (n) (ntree-children n)) nodes) (1+ depth)))))))))

(document! ntree-traverse-or-find "order:  'IN_ORDER 'PRE_ORDER 'POST_ORDER or 'LEVEL_ORDER.
- G_IN_ORDER visits a node's left child first, then the node itself, then its right child.
This is the one to use if you want the output sorted according to the compare function. The
inorder traversal of an N-ary tree is defined as visiting all the children except the last
then the root and finally the last child recursively
- G_PRE_ORDER visits a node, then its children.
- G_POST_ORDER visits the node's children, then the node itself.
- G_LEVEL_ORDER is not implemented for Balanced Binary Trees. For N-ary Trees, it vists the
root node first, then its children, then its grandchildren, and so on. Note that this is
less efficient than the other orders.

flags:  'TRAVERSE_ALL 'TRAVERSE_LEAVES and 'TRAVERSE_NON_LEAVES

max-depth: the maximum depth of the traversal. Nodes below this depth will not be visited.
If max_depth is -1 all nodes in the tree are visited. If depth is 1, only the root is visited.
If depth is 2, the root and its children are visited. And so on.
```

```
func: the function called for each node. Parameters: (node deep)
if the function returns #t, the evaluation stops and the node is returned
"
)
(define (i-build-node-list root flags max-depth func) ;;post order, per ora
  (define (nst r depth)
    (if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
'()
(if (null? (ntree-children r))
    (begin
      (if (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL))
  (cons (func r depth) '())
  '()))
    (begin
      (append
       (append-map (lambda (it) (nst it (1+ depth))) (ntree-children r))
       (if (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
   (list (func r depth))
   '()))))))
  (nst root 0)
  )
;;
(define-public (ntree-traverse-map root order flags max-depth func)
  "Traverses the nary tree and builds a list with the results of func (node depth) applied
to the traversed nodes.
order:  'IN_ORDER 'PRE_ORDER 'POST_ORDER or 'LEVEL_ORDER.
flags:  'TRAVERSE_ALL 'TRAVERSE_LEAVES and 'TRAVERSE_NON_LEAVES"
  (let ((result '()))
    (cond
     ((eqv? order 'IN_ORDER)
      (let loop ((r root) (depth 0))
(if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
    '()
    (if (and (null? (ntree-children r))
   (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL)))
(mtfa-m-cons (func r depth) result)
(begin ;;visit all children except the last
  (for-each (lambda (it) (loop it (1+ depth))) (drop-right (ntree-children r) 1))
  (when (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
    (mtfa-m-cons (func r depth) result ))
  (loop (last (ntree-children r)) (1+ depth)))))))
     ;;
     ((eqv? order 'PRE_ORDER)
      (let loop ((r root) (depth 0))
(if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
    '()
    (if (and (null? (ntree-children r))
   (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL)))
(mtfa-m-cons (func r depth) result)
(begin
  (when  (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
    (mtfa-m-cons (func r depth) result))
  ;;visit all children
  (for-each (lambda (it) (loop it (1+ depth))) (ntree-children r)))))))
     ;;
     ((eqv? order 'POST_ORDER)
      (set! result (i-build-node-list root flags max-depth func))
      ;; (let loop ((r root) (depth 0))
      ;;  (if (or (null? r) (and (>= max-depth 0) (>= depth max-depth)))
      ;;      '()
```

```
      ;;      (if (and (null? (ntree-children r))
      ;;     (or (eqv? flags 'TRAVERSE_LEAVES)(eqv? flags 'TRAVERSE_ALL)))
      ;;  (mtfa-m-cons (func r depth) result)
      ;;  (begin
      ;;    (for-each (lambda (it) (loop it (1+ depth))) (ntree-children r))
      ;;    (when  (or (eqv? flags 'TRAVERSE_NON_LEAVES)(eqv? flags 'TRAVERSE_ALL))
      ;;      (mtfa-m-cons (func r depth) result ))))))
      )
     ;;
     ((eqv? order 'LEVEL_ORDER)
      (if (null? root)
  '()
  (let loop ((nodes (list root))
     (depth 0))
    (if (null? nodes)
'()
(begin
  (for-each (lambda (n)
      (cond
       ((eqv? flags 'TRAVERSE_ALL)
(mtfa-m-cons (func n depth) result))
       ((and (eqv? flags 'TRAVERSE_LEAVES) (null? (ntree-children n)))
(mtfa-m-cons (func n depth) result))
       ((and (eqv? flags 'TRAVERSE_NON_LEAVES) (not (null? (ntree-children n))))
(mtfa-m-cons (func n depth) result))))
    nodes)
  (loop (append-map (lambda (n) (ntree-children n)) nodes) (1+ depth))))))))
    result))
;;
(define-public (ntree-traverse-fold root order flags max-depth func init)
  "Traverses the nary tree and builds as the fold function (node depth prec) applied to the
traversed nodes.
order:  'IN_ORDER 'PRE_ORDER 'POST_ORDER or 'LEVEL_ORDER.
flags:  'TRAVERSE_ALL 'TRAVERSE_LEAVES and 'TRAVERSE_NON_LEAVES"
  (let ((prec init)
(lst (ntree-traverse-map root order flags max-depth (lambda (c d) (cons c d)))))
    (fold (lambda (c p)
    (func (car c) (cdr c) p))
  init
  lst)))
;;
(defun-public ntree-children-foreach-or-find (root flags func)
  (for-each (lambda (n)
      (cond
       ((eqv? flags 'TRAVERSE_ALL)
(when (func n) (return n)))
       ((and (eqv? flags 'TRAVERSE_LEAVES) (null? (ntree-children n)))
(when (func n) (return n)))
       ((and (eqv? flags 'TRAVERSE_NON_LEAVES) (not (null? (ntree-children n))))
(when (func n) (return n)))))
    (ntree-children root)))
(document! ntree-children-foreach-or-find "flags:  'TRAVERSE_ALL 'TRAVERSE_LEAVES and 'TRAVERSE_NO
func: the function called for each node. Parameters: (node). If the function returns #t,
stops and returns the current child
")
;;
(define-public (ntree-children-map root flags func)
  "Applies func to each child satisfying traversing flags, returns a list"
  (map (lambda (n)
 (cond
  ((eqv? flags 'TRAVERSE_ALL) (func n))
```

```
  ((and (eqv? flags 'TRAVERSE_LEAVES) (null? (ntree-children n))) (func n))
  ((and (eqv? flags 'TRAVERSE_NON_LEAVES) (not (null? (ntree-children n)))) (func n))
  (#t '())))
       (ntree-children root)))
;;
(define-public (ntree-children-fold root flags func init)
  "Applies func (current prec) to each child satisfying traversing flags, returns the fold"
  (fold (lambda (curr prec)
 (cond
  ((eqv? flags 'TRAVERSE_ALL) (func curr prec))
  ((and (eqv? flags 'TRAVERSE_LEAVES) (null? (ntree-children curr))) (func curr prec))
  ((and (eqv? flags 'TRAVERSE_NON_LEAVES) (not (null? (ntree-children curr)))) (func curr
prec))
  (#t prec)))
init
(ntree-children root)))
;;
(defun-public ntree-parents-foreach-or-find (root func distance)
  (let loop ((root root) (func func) (dist 0))
    (if (or (null? root) (null? root) (> dist distance))
'()
(begin
  (when (func root dist)
    (return root))
  (loop (ntree-parent root) func (1+ dist))))))
(document! ntree-parents-foreach-or-find "func: the function called for each node. Parameters:
(node distance). If the function returns #t, stops and returns the current ancestor")
;;
(define-public (ntree-parents-map root func distance)
  "Applies func to each ancestor, returns a list"
  (let loop ((root root) (func func) (dist 0))
    (if (or (null? root) (null? root) (> dist distance))
'()
(cons (func root dist) (loop (ntree-parent root) func (1+ dist))))))
;;
(define-public (ntree-parents-fold root func init distance)
  "Applies func (current prec dist) to each ancestor until the distance is less than distance"
  (let loop ((root root) (func func) (dist 0)  (prec init))
    (if (or (null? root) (null? root) (> dist distance))
prec
(loop (ntree-parent root) func (1+ dist) (func root prec dist)))))
;;
(define-public (ntree-get-root node)
  (if (null? node)
      '()
      (if (null? (ntree-parent node))
  node
  (ntree-get-root (ntree-parent node)))))
;;
(define-public (ntree-child-index node)
  "Returns the position of the node in the children list of the parent"
  (if (or (null? node) (null? (ntree-parent node)))
      '()
      (list-index (cut eqv? node <>) (ntree-children (ntree-parent node)))))
;;
(define-public (ntree-first-child node)
  (if (or (null? node) (null? (ntree-children node)))
      '()
      (car (ntree-children node))))
;;
(define-public (ntree-last-child node)
```

```
  (if (or (null? node) (null? (ntree-children node)))
      '()
      (last (ntree-children node))))
;;
(define-public (ntree-nth-child node index)
  (if (or (null? node) (null? (ntree-children node)))
      '()
      (list-ref (ntree-children node) index)))
;;
(define-public (ntree-first-sibling node)
  (if (null? node)
      '()
      (if (null? (ntree-prev node))
  node
  (ntree-first-sibling (ntree-prev node)))))
;;
(define-public (ntree-last-sibling node)
  (if (null? node)
      '()
      (if (null? (ntree-next node))
  node
  (ntree-last-sibling (ntree-next node)))))
;;
(define-public (ntree-next-sibling node)
  (if (null? node)
      '()
      (ntree-next node)))
;;
(define-public (ntree-prev-sibling node)
  (if (null? node)
      '()
      (ntree-prev node)))
;;
(define-public (ntree-is-leaf n)
  (and (not (null? n)) (null? (ntree-children n))))
(define-public (ntree-is-root n)
  (and (not (null? n)) (null? (ntree-parent n))))
;;
(define-public (ntree-depth n)
  (if (null? n)
      0
      (1+ (ntree-depth (ntree-parent n)))))
;;
(define-public (ntree-nodes n flags)
  "The flags are TRAVERSE_ALL, TRAVERSE_LEAVES and TRAVERSE_NON_LEAVES"
  (let ((count 0))
    (ntree-traverse-or-find n 'PRE_ORDER flags -1 (lambda (n d) (set! count (1+ count)) #f))
    count))
;;
(define-public (ntree-n-children n)
  (if (null? n)
      0
      (length (ntree-children n))))
;;
(define-public (ntree-is-ancestor n d)
  (if (or (null? n) (null? d))
      #f
      (if (eqv? n (ntree-parent d))
  #t
  (ntree-is-ancestor n (ntree-parent d)))))
;;
```

```
(define-public (ntree-max-height n)
  (if (null? n)
      0
      (let ((hmax 0))
(ntree-traverse-or-find n 'PRE_ORDER 'TRAVERSE_ALL -1 (lambda (n d) (when (< hmax d) (set!
hmax d)) #f))
(1+ hmax))))
;;
(define-public (ntree-unlink n)
  (if (null? n)
      (values '() '())
      (if (null? (ntree-parent n))
  (values n '())
  (let ((root (ntree-get-root n))
(parent (ntree-parent n))
(prev (ntree-prev n))
(next (ntree-next n)))
    ;;lo rendo root!
    (ntree-parent! n '())
    (ntree-prev! n '())
    (ntree-next! n '())
    ;;
    (if (and (null? prev) (null? next))
(ntree-children! parent '()) ;;era figlio unico, lo tolgo
(if (null? prev)
    (begin
      (ntree-children! parent (cdr (ntree-children parent)))
      (ntree-prev! next '()))
    (if (null? next)
(begin
  (ntree-children! parent (drop-right (ntree-children parent) 1))
  (ntree-next! prev '()))
(begin
  (ntree-children! parent (delv n (ntree-children parent)))
  (ntree-next! prev next)
  (ntree-prev! next prev)))))
    (values root n)))))
;;
;;
;;Some test
(define-public (ntree-print n)
  (Show! (map (lambda (i)
  (list (ntree-data i)
(ifnot (null? (ntree-prev i))
       (ntree-data (ntree-prev i))
       '())
(ifnot (null? (ntree-next i))
       (ntree-data (ntree-next i))
       '())))
(ntree-children n))))
(define-public (ntree-print-all n)
  (ntree-traverse-or-find n 'LEVEL_ORDER 'TRAVERSE_ALL -1 (lambda (n d) (display (ntree-data
n)) (display " ") #f)) (newline))
;;
(define-public (ntree-build degree nodes)
  (let ((idx -1))
    (let loop ((degree degree) (nodes nodes))
      (if (> nodes 1)
  (let ((nch (max 1 (min nodes (remainder (mtfa-rand-ui) degree))))
(root (ntree-new (mtfa-m-add idx))))
    (do ((i 0 (1+ i))) ((> i nch))
```

```
      (let ((ch (loop degree (/ nodes (1+ nch)))))
(if (not (null? ch))
    (ntree-insert root -1 ch))))
    root)
  (ntree-new (mtfa-m-add idx))))))
;;

(define (ntree-test)
  (define root (ntree-new 0))
  (define ch1 (ntree-new 1))
  (define ch2 (ntree-new 2))
  (define ch3 (ntree-new 3))
  (define ch4 (ntree-new 4))
  (define ch5 (ntree-new 5))
  (define ch6 (ntree-new 6))
  (define ch7 (ntree-new 7))
  (define ch8 (ntree-new 8))
  (define ch9 (ntree-new 9))
  (define ch10 (ntree-new 10))
  (define ch11 (ntree-new 11))
  (define ch12 (ntree-new 12))
  (define ch13 (ntree-new 13))
#|
                    0
           1        2        3
         4  5       6     7  8  9  10
             11   12            13
|#
  (ntree-insert root -1 ch1)
  (ntree-insert root -1 ch2)
  (ntree-insert root -1 ch3)

  (ntree-insert ch1 -1 ch4)
  (ntree-insert ch1 -1 ch5)
  (ntree-insert ch2 -1 ch6)
  (ntree-insert ch3 -1 ch7)
  (ntree-insert ch3 -1 ch8)
  (ntree-insert ch3 -1 ch9)
  (ntree-insert ch3 -1 ch10)
  (ntree-insert ch5 -1 ch11)
  (ntree-insert ch6 -1 ch12)
  (ntree-insert ch9 -1 ch13)

  ;; (ntree-insert root -1 ch4)
  ;; (print root)
  ;; (display "(ntree-insert root 2 ch5)\n")
  ;; (ntree-insert root 2 ch5)
  ;; (print root)
  ;; (display "(ntree-insert root 1 ch6)\n")
  ;; (ntree-insert root 1 ch6)
  ;; (print root)
  ;; (display "(ntree-insert root 5 ch7)\n")
  ;; (ntree-insert root 5 ch7)
  ;; (print root)
  ;; (display "(ntree-insert root 0 ch8)\n")
  ;; (ntree-insert root 0 ch8)
  ;; (print root)
  ;; (display "(ntree-insert-before root ch5 ch9)\n")
  ;; (ntree-insert-before root ch5 ch9)
  ;; (print root)
  ;; (display "(ntree-insert-after root ch5 ch10)\n")
```

```
;; (ntree-insert-after root ch5 ch10)
;; (print root)
;; (display "(ntree-reverse-children root)\n")
;; (ntree-reverse-children root)
(ntree-print root)

)
```

3.15 module Ethereum

A module for blockchain management

- `mtfa-eth-to-wei (#:optional (amount 0) (currency ""))`
  `mtfa-eth-from-wei (#:optional (amount 0) (currency ""))`
  `mtfa-eth-empty-address "0000000000000000000000000000000000000000`
  `(mtfa-eth-build-infura-req method params id)`
  `mtfa-eth-get-infura (net path method #:optional (params #()) (id 1)`
  `mtfa-eth-scmjson-to-hexrlp (l)`
  `mtfa-eth-hexrlp-to-scmjson (hexrlp)`
  `mtfa-eth-build-transaction (nonce gasPrice gasLimit toAddress value data)`
  `mtfa-eth-build-signed-transaction (trans network hex-prk)`
  `mtfa-eth-get-ethereum-address (prk)`
  `mtfa-eth-infura-gasPrice (net path)`
  `mtfa-eth-infura-estimateGas (net path from to gasPrice value hdata)`
  `mtfa-eth-infura-getBalance (net path who)`
  `mtfa-eth-infura-getTransactionCount (net path who latest-earliest-pending)`
  `mtfa-eth-infura-sendRawTransaction (net path signed)`
  `mtfa-eth-infura-getTransactionReceipt (net path trans-hash)`
  `mtfa-eth-transfer (net path user1 user1-prk user2 howmuch)`

Chapter 4

The VAPP - Virtual Application container

To download the container and its updates you have to login into hesperides docker hub and pull or run the Virtual Application container.
The simplest way is

- docker login
- `<username> h019`
- `<password> H23aq,!09Uh`
- docker pull h019/va:latest

Now, to test if all is ok, on a system with cuda installed:
`docker run -ti --privileged --network=host --gpus="all" --volume "$PWD":/vapps -e "DISPLAY=${DISPLAY:-:`
`-v /tmp/.X11-unix:/tmp/.X11-unix:rw -v /:/altro h019/va nvidia-smi` To check if it is ok, on a system without cuda installed:
`docker run -ti --network=host --volume "$PWD":/vapps -v /:/other h019/va /bin/bash -i`

4.1 A simple run of the VAPP - Virtual Application

This is a simple WEB service exposing some APIs.

4.1.1 The file example.conf

```
LbUser_VA_VRules = example.ars LbUser_VA_VReports = reports LbUser_OutIp = 0.0.0.0 LbUser_AsDaemon=false
LbUser_Id_WhoAmI=VA LbUser_Initialize=127.0.0.1:61100 LbUser_SVA_Listen= LbUser_FromVACommands=
LbUser_Hy_VA_Sisters= LbUser_Hy_KAM_Children= LbUser_SecureVa= LbUser_VAObservedApps= LbUser_VA_GetAllHTT
LbUser_VA_QueryFilters= LbUser_VA_HwAgentPort= LbUser_VA_HwAgentIp= LbUser_VA_SwAgentPort= LbUser_VA_SwAg
LbUser_VA_AskForUrl= LbUser_VA_LocalSH_on = true LbUser_VA_TraceDomain=127.0.0.1:61107 LbUser_VA_KAM_Comm
LbUser_VA_KAM_CommandIp=127.0.0.1 LbUser_ListenIp=0.0.0.0 LbUser_ListenPort=19999 LbUser_VListenIpPort=
LbUser_VTCPPort= LbUser_VSSLPort= LbUser_HowManyInstances=8 LbUser_ThPoolSize=16 LbUser_LspCode=example.s
LbUser_LogFile= LbUser_NewCall=true LbUser_Rules= LbUser_BlockMessage=
```

4.1.2 The file example.ars

```
define urlset url_srv = { /msvc };
define ipset ip_all = { *.*.*.*:* };

DEFINE AR "AR-MICROSVC"
  CONDITION
  http.url is in url_srv
  ACTION
  MANAGE "API"
;
```

```
DEFINE AR "AR-HOLE"
CONDITION
net.ipdst is in ip_all
ACTION
ANSWER "ok"
;
```

4.1.3 The file example.scm

```
(use-modules
 (mtfa error-handler)
 (mtfa utils)
 (mtfa serializer)
 (mtfa unordered-set)
 (mtfa unordered-map)
 (mtfa star-map)
 (mtfa simple_db)
 (mtfa certs)
 (mtfa eis)
 ;;(mtfa fsm)
 (mtfa va)
 (mtfa extset)
 (mtfa umset)
 (mtfa web)
 (mtfa brg)
 (mtfa lazy-seq)

 ;;
 (srfi srfi-1)
 (srfi srfi-9)
 (srfi srfi-11)
 ;;(srfi srfi-18)
 (srfi srfi-19)
 (srfi srfi-26)
 ;;(srfi srfi-28)
 (srfi srfi-43)
 (srfi srfi-60)
 (web uri)
 (system foreign)

 (rnrs bytevectors)
 (rnrs arithmetic bitwise)
 ((rnrs io ports)
  #:select (string->bytevector bytevector->string)
  #:prefix ioports:)

 (ice-9 format)
 (ice-9 ftw)
 (ice-9 rdelim)
 (ice-9 pretty-print)
 (ice-9 regex)
 (ice-9 iconv)
 (ice-9 peg)
 (ice-9 peg string-peg)
 (ice-9 vlist)
 (ice-9 q)
 (ice-9 binary-ports)
 (ice-9 threads)
 (ice-9 hash-table)
 (ice-9 control)
 (ice-9 match)
```

```
 (oop goops)
 (oop goops describe)
 (sxml simple)
 (sxml ssax)
 (sxml xpath)
 (json)
 (system syntax)

 ;;
 ;;
 )


(mtfa-rand-seed (string->number (TimeStamp)))


(define (ConvertToString v)
  (with-output-to-string (lambda () (display v))))

(define users '(
 ("user01" . "user_01")
 ("user02" . "user_02")
 ("user03" . "user_03")
 ("user04" . "user_04")
 ("user05" . "user_05")
 ("user06" . "user_06")
 ))

(defun Manage::API (actionl pbuf)
  (return (eis::GiveErrorHTTP401)) =>
  (let* ((pars
  (sort
   (map (lambda (v) (string-split v #\=))
(string-split (mtfa-eis-get-current-pars pbuf) #\&))
   (lambda (v1 v2) (string< (car v1) (car v2)))))
 (result #f))
    (Show "PARS: " pars)
    (match pars
      (
       (("cmd" "add") ("p1" v1) ("p2" v2) ("username" username))
       (set! result (+ (string->number v1) (string->number v2)))
       )
      ((("cmd" "sub") ("p1" v1) ("p2" v2) ("username" username))
       (set! result (- (string->number v1) (string->number v2))))
      ((("cmd" "mul") ("p1" v1) ("p2" v2) ("username" username))
       (set! result (* (string->number v1) (string->number v2))))
      ((("cmd" "div") ("p1" v1) ("p2" v2) ("username" username))
       (set! result (/ (string->number v1) (string->number v2))))
      ((("cmd" "sqrt") ("p1" v1) ("username" username))
       (set! result (sqrt (string->number v1))))
      ((("cmd" "login") ("p1" username) ("p2" password))
       (set! result (assoc-ref users username))
       (when result
 (set! result (string=? password result))))
     (_ (set! result #f))
     )

    (eis::GiveHTTPHtmlAnswer  (string-append "result=" (ConvertToString result)))
   ))
```

```
(eis::function-pointer-add "API" Manage::API)
```

4.1.4 How to run the service

```
docker run -ti --network=host --volume "$PWD":/vapps h019/va /va example.conf\\

Then, request the service using the simple command line:
curl "http://127.0.0.1:19999/msvc?cmd=add&p1=10&p2=20&username=user01"

The result given is "result=30"
```