

École nationale supérieure des mines de Nantes  
ASCOLA Research Group



# AccLab User Guide

Version 1.0

June 3, 2015

Walid BENGHABRIT

Supervisor : **Prof. Jean-Claude Royer**  
Co-Supervisor : **A/Prof. Hervé Grall**

# Contents

Table of contents . . . . .	1
1 Getting started . . . . .	2
1.1 Install AccLab . . . . .	2
1.2 Using AAL compiler "aalc" . . . . .	2
1.3 Writing your first AAL program . . . . .	2
1.4 Running the program . . . . .	3
1.5 Using core libraries . . . . .	4
1.6 Using the shell . . . . .	6
2 AAL language . . . . .	8

# 1 Getting started

## 1.1 Install AccLab

- To use the ltl prover, you need to put the following executable files (tspass, fotl-translate) in tools/your\_platform/ (linux/mac/win)
- Basic run : python aalc.py (you need python3.4.0 or greater)
- Run an AAL file : python aalc.py -i testfile.aal

## 1.2 Using AAL compiler "aalc"

Listing 1: aalc options

```
aalc
-h  --help          display this help and exit
-i  --input         the input file
-o  --output        the output file
-c  --compile       compile the file, that can be loaded after using -l
-m  --monodic       apply monodic check on aal file
-s  --shell         run a shell after handling aal program
-k  --check         perform a verbose check
-l  --load          load a compiled aal file (.aalc) and run a shell
-t  --fotl          translate the aal program into FOTL
-r  --reparse       reparsing tspass file
-r  --recompile     recompile the external files
-b  --no-colors     disable colors in output
-x  --compile-stdlib compile the standard library
-d  --hotswap       enable hotswapping (for development only)
-a  --ast           show ast tree
```

## 1.3 Writing your first AAL program

Let consider the following scenario, we have three actors :

- cloud storage service : let call it **css** which is a cloud service provider
- alice and bob : an end users that uses css service

The **css** offers the following services : read (a user reads some data from css server), store (a user stores some data into css server), delete (a user deletes some data from css server). **css** allows users to read/store/delete only their data on his server, and don't allow them to read other customers data. **css** can also read and delete any data from his server.

Alice want to check if **css** policy respect her privacy. Typically she want to know if she is allowed to performs some actions and if bob can read here data.

a. **Declaring services** The services are the

```
SERVICE read
SERVICE store
SERVICE delete
```

b. **Declaring actors** : first we need to declare our actors

## 1. GETTING STARTED

---

```
// Agents declaration
AGENT alice
AGENT bob
AGENT css
```

- c. **Linking services and actors** First we need to declare our actors

```
AGENT alice TYPES() REQUIRED(read store delete) PROVIDED()
AGENT bob   TYPES() REQUIRED(read store delete) PROVIDED()
AGENT css   TYPES() REQUIRED() PROVIDED(read store delete)
```

- d. **Defining policies** First we need to declare our actors

```
/*
 * Cloud storage service provider policy
 */
CLAUSE css_policy (
  FORALL d:data FORALL a:Actor

  // Allow users to read their data
  IF (d.subject == a) THEN {
    PERMIT a.read[css](d)
  } AND

  // Deny access to read other
  IF (d.subject != a) THEN {
    DENY a.read[css](d)
  } AND

  // Allow css to read/delete stored data
  PERMIT css.read[css](d) AND
  PERMIT css.delete[css](d)
)

/*
 * Alice's preferences
 */
CLAUSE alice_policy (
  FORALL d:data
  // Alice want to be able to read all her data stored on css
  IF (d.subject == alice) THEN {
    PERMIT alice.read[css](d)
  }
)
```

- e. **Writing checks** Now we want to check if Alice's privacy preferences are respected by the `css` policy. To do this, we can call the macro `validate` and passing the the clauses names as arguments. Important : Note that the order of arguments is important.

```
CALL validate("css_policy" "alice_pref")
```

### 1.4 Running the program

- Run the AAL program

```
root@root/:$ python aalc -i examples/tuto0.aal
```

```
----- Monodic check -----
```

## 1. GETTING STARTED

---

```
Monodic check passed !
----- Starting Validity check -----
c1 : css_policy
c2 : alice_pref
----- Checking c1 & c2 consistency :
-> Satisfiable
----- Checking c1 => c2 :
-> Satisfiable
----- Checking ~(c1 => c2) :
-> Unsatisfiable

[VALIDITY] Formula is valid !
----- Validity check End -----

File : examples/tuto0.aal

Execution time : 0.24277639389038086
```

Here the result of

- Perform an detailed check

```
root@root/:$ python aalc -i examples/tuto0.aal -k

----- Start Checking -----

** DECLARATIONS
[DECLARED AGENTS] : 3
[DECLARED SERVICES] : 6
[DECLARED DATA] : 0
[DECLARED TYPES] : 10

*** Forwards references check
[AGENTS] : 0
[SERVICES] : 0
[DATA] : 0
[TYPES] : 0

** LOADED libraries
[LIBS] : 2

** CLAUSES
[CLAUSES] : 2
Monodic test :
|css_policy | Formula is monodic ! |
|alice_pref | Formula is monodic ! |
----- Checking End -----
```

- Perform monodic test on all clauses :

```
root@root/:$ python aalc -i examples/tuto0.aal -m

----- Start Checking -----
|css_policy | Formula is monodic ! |
|alice_pref | Formula is monodic ! |
----- Checking End -----
```

- Translate AAL program into FOTL (in tpass syntax):

## 1. GETTING STARTED

---

```
root@root/:$ python aalc -i examples/tuto0.aal -t

----- FOTL Translation start -----

%%% START EVN %%%
%%% Types knowledge
data(CTS) &
Actor(CTS) &
DataSubject(CTS) &
DataController(CTS) &
DataProcessor(CTS) &
DwDataController(CTS) &
Auditor(CTS) &
CloudProvider(CTS) &
CloudCustomer(CTS) &
EndUser(CTS) &

%%% Action authorizations
(![x, y, z] (read(x, y, z) => Pread(x, y, z))) &
(![x, y, z] (store(x, y, z) => Pstore(x, y, z))) &
(![x, y, z] (delete(x, y, z) => Pdelete(x, y, z))) &
(![x, y, z] (read(x, y, z) => Pread(x, y, z))) &
(![x, y, z] (write(x, y, z) => Pwrite(x, y, z))) &
(![x, y, z] (audit(x, y, z) => Paudit(x, y, z))) &

%%% Actors knowledge
Actor(alice) &
Actor(bob) &
Actor(css) &

%%% END EVN %%%

%% Clause : css_policy
always(![d] ( data(d) & (![a] ( Actor(a) & ((( (subject(d, a) => Pread(a, css, d)) &
(~subject(d, a) => ~Pread(a, css, d))) & Pread(css, css, d)) & Pdelete(css, css, d)))) )
%% Clause : alice_pref
always(![d] ( data(d) & ( (subject(d, alice) => Pread(alice, css, d)))) )

----- FOTL Translation end -----
```

### 1.5 Using core libraries

You can load external AAL files using `LOAD "aal_file"`(without the extension)

**core.types** Contains the basic types declarations (Actor, DataSubject, DataController, DataProcessor, ...)

```
LOAD "core.types"
```

**core.macros** Contains some useful macros.

```
// Loading lib
LOAD "core.macros"
```

### 1.6 Advanced checks

dzd

### 1.7 Using the shell

The shell is a useful tool for developing

- Run the shell.

```
aalc -i examples/tuto1.aal -s
```

```
/* Result :  
shell >
```

```
*/
```

- Type help to show the shell help.

```
Shell Help
```

```
- call(macro, args)  call a macro where /  
                    *macro : is the name of the macro  
                    *args : a list of string; << ex : ['args1', 'args2', ...'argsN'] >>  
- clauses()          show all declared clauses in the loaded aal program  
- macros()           show all declared macros in the loaded aal program  
- quit / q           exit the shell  
- help / h / man()   show this help  
- self              the current compiler instance of the loaded aal program  
- aalprog            the current loaded aal program  
- man(arg)           print the help for the given arg  
- hs(module)         hotswaping : reload the module  
- r()               hot-swaping the shell
```

- Here an example, we print all clauses in the AAL program.

```
shell> clauses()
```

```
/* Result :  
kim_policy cloudX_policy
```

```
*/
```

- self variable represent the co

```
shell> self
```

```
/* Result :  
<AALCompiler.AALCompilerListener object at 0x7f8b00ce8630>
```

```
*/
```

```
shell> man(self)
```

```
/* Result :  
printing manual for <class 'AALCompiler.AALCompilerListener'>  
Manual for aal compiler visitor  
- Attributes  
  - aalprog      Get the AAL program instance
```

## 1. GETTING STARTED

---

```
- file          The AAL source file
- libs          Show the loaded libraries
- libsPath      Print the standard lib path
- Methods
- load_lib(lib_name)  Load an aal file
- clause(clauseId)    Get a clause
- show_clauses()      Print all clauses
- get_clauses()       Get all clauses (array format)
- get_macros()        Get all macros

*/

shell> man(aalprog)

/* Result :
printing manual for <class 'AALMetaModel.m_aalprog'>

AAL program class.
Note that clauses and macros extends a declarable type, but are not in the declarations dict

Attributes
- clauses: a list that contains all program clauses
- declarations: a dictionary that contains lists of typed declarations
- comments: a list that contains program's comment
- macros: a list that contains program's macros declarations
- macroCalls: a list that contains program's comment

*/
```

- hotswaping commands are used for debugging purpose only. `r()` command allows you to reload the shell after  
`hs(module)` reloading other modules after ! IMORTANT : to use hotswaping properly you must enable it explicitly in `aalc` arguments `-d / -hotswap`,



## 2 AAL language

Listing 2: AAL Syntax

```
// AAL CORE
AALprogram ::= (Declaration | Clause | Comment | Macro | MacroCall | Loadlib
               | LtlCheck | CheckApply | Exec | Behavior)
Declaration ::= AgentDec | ServiceDec | DataDec | TypesDec | varDec
AgentDec    ::= AGENT Id [TYPES '(' Type '*'')' REQUIRED '(' service* ')' PROVIDED '(' service* ')']
ServiceDec  ::= SERVICE Id [TYPES '(' Type* ')'] [PURPOSE '(' Id* ')']
DataDec     ::= DATA Id TYPES '(' Type* ')' [REQUIRED '(' service* ')' PROVIDED '(' service* ')']
SUBJECT agent
VarDec      ::= Type_Id Id [attr_Id '(' value* ')']*
Clause      ::= CLAUSE Id '(' [Usage] [Audit Rectification] ')'
Usage       ::= ActionExp
Audit       ::= AUDITING Usage
Rectification ::= IF_VIOLATED_THEN Usage
ActionExp   ::= Action | NOT ActionExp | Modality ActionExp | Condition
               | ActionExp (AND|OR|ONLYWHEN|UNTIL|UNLESS) ActionExp
               | Author | Quant* | IF '(' ActionExp ')' THEN '{' ActionExp '}'
Exp         ::= Variable | Constant | Variable.Attribute
Condition   ::= [NOT] Exp | Exp ['==' | '!='] Exp | Condition (AND|OR) Condition
Author      ::= (PERMIT | DENY) Action
Action      ::= agent.service ['[agent]'] '(' Exp ')' [Time] [Purpose]
Quant       ::= (FORALL | EXISTS) Var [WHERE Condition]
Variable    ::= Var ':' Type
Modality    ::= MUST | MUSTNOT | ALWAYS | NEVER | SOMETIME
Time        ::= (AFTER | BEFORE) Date | Time (AND | OR) Time
Date        ::= STRING
Type, var, val, attr Id, agent, Constant, Purpose ::= literal

// AAL Type extension
TypesDec    ::= TYPE Id [EXTENDS '(' Type* ')'] ATTRIBUTES '(' AttributeDec* ')' ACTIONS '('
               ActionDec* ')'
AttributeDec ::= Id ':' Type
ActionDec    ::= Id
Type, Id     ::= literal

// Reflexion extension
Macro       ::= MACRO Id '(' param* ')' '(' mcode ')'
MCode       ::= """_Meta_model_api+_Python3_code_(subset)"""
MCall       ::= CALL Id '(' param* ')'
LoadLib     ::= LOAD STRING;
Exec        ::= EXEC MCode

// FOTL checking extension
LtlCheck    ::= CHECK Id '(' param* ')' '(' check ')'
check       ::= FOTL_formula + clause(Id) [.ue | .ae | .re]
CheckApply  ::= APPLY Id '(' param* ')'

// Behavior extension
Behavior    ::= BEHAVIOR Id '(' ActionExp ')'
```

### **3 Using AccLab web GUI**

TODO

Draft