Appendix

A Language Specification

A.1 Policy Type

```
abstract_base_policy =
    "begin" "abstract-base-policy" string ":"
        "begin" "inheritance" ":"
            inheritance_section
        "end" "inheritance" ";"
        "begin" "components" ":"
            components_section
        "end" "components" ";"
        "begin" "conditions" ":"
            conditions_section
        "end" "conditions" ";"
        "begin" "primitives" ":"
            primitives_section
        "end" "primitives" ";"
    "end" "abstract-base-policy" ";"
    ;

inheritance_section =
    /* empty */
    | inheritance_section inheritance
    ;

inheritance =
    "begin" string ":"
        "all"
    "end" ";"
    | "begin" string ":"
        "components" ":" inheritance_item_list ";"
        "conditions" ":" inheritance_item_list ";"
        "primitives" ":" inheritance_item_list ";"
    "end" ";"
    ;

inheritance_item_list =
    "all"
    | "{" "}"
    | "{" inheritance_item_list_nonempty "}"
    ;

inheritance_item_list_nonempty =
    inheritance_item_list_nonempty "," string "as" string
    | inheritance_item_list_nonempty "," string
    | string "as" string
    | string
    ;

components_section =
    components_section component_definition
    | component_definition
    ;

component_definition =
    "set" string ";"
    | "relation" string "(" string_list_two_elem ")" ";"
    | "mapping" string "(" string_list_nonempty ":"
        string ")" ";"
    | "mapping" string "(" string_list_nonempty ":"
        "2" "^" string ")" ";"
    ;

conditions_section =
    /* empty */
    | conditions_section condition_definition
    ;

condition_definition =
    string "(" parameter_list ")" ":" condition ";"
    ;

condition =
    comparable "==" comparable
    | comparable "!=" comparable
    | comparable "in" string_or_function
    | comparable "not" "in" string_or_function
    | "query" "(" value "," "[" string_list "]" ")"
    | string "(" string_list_nonempty ")"
    | "(" condition ")"
    | "not" condition
    | condition "and" condition
    | condition "or" condition
    | "forall" string "in" string_or_function ":" condition
    | "exists" string "in" string_or_function ":" condition
    | "forall" tuple "in" string ":" condition
    | "exists" tuple "in" string ":" condition
    ;
```

```
string_or_function =
    string
    | string "(" string_list_nonempty ")"
    ;

comparable =
    string_or_function
    | tuple
    ;

parameter_list =
    /* empty */
    | parameter_list_nonempty
    ;

parameter_list_nonempty =
    parameter_list_nonempty "," string string
    | parameter_list_nonempty "," "2" "^" string string
    | string string
    | "2" "^" string string
    ;

primitives_section =
    primitives_section primitive_definition
    | primitive_definition
    ;

primitive_definition =
    "begin" string "(" parameter_list_nonempty ")" ":"
        statement_list_nonempty
    "end" ";"
    ;

statement_list_nonempty =
    statement_list_nonempty for_statement
    | statement_list_nonempty statement
    | for_statement
    | statement
    ;

for_statement =
    "begin" "for" string "in" string ":"
        statement_list_nonempty
      "end" "for" ";"
    | "begin" "for" tuple "in" string ":"
        statement_list_nonempty
      "end" "for" ";"
    ;

statement =
    string "=" string "+" string ";"
    | string "=" string "-" string ";"
    | string "=" string "+" "{" string "}" ";"
    | string "=" string "-" "{" string "}" ";"
    | string "=" string "+" "{" tuple "}" ";"
    | string "=" string "-" "{" tuple "}" ";"
    | string "=" string "+" "{" "(" string_list_nonempty
        ":" string ")" "}" ";"
    | string "=" string "-" "{" "(" string_list_nonempty
        ")" "}" ";"
    | string "=" string "+" "{" "(" string_list_nonempty
        ":" "{" string_list_nonempty "}" ")" "}" ";"
    | string "=" string "(" string_list_nonempty ")" ";"
    | string "(" string_list_nonempty ")" ";"
    ;

tuple =
    "[" string_list_nonempty "]"
    ;

value =
    "'" string "'"
    ;

string_list =
    /* empty */
    | string_list_nonempty
    ;

string_list_nonempty =
    string_list_nonempty "," string
    | string
    ;

string_list_two_elem =
    string_list_two_elem "," string
    | string "," string
    ;

string =
    letter sub_string
    ;

sub_string =
    /* empty */
    | sub_string letter
    | sub_string digit
    | sub_string symbol
    ;
```

```
letter = "a" | ... | "z" | "A" | ... | "Z" ;

digit = "0" | ... | "9" ;

symbol = "-" | "_" ;
```

A.2 Policy

```
policy =
    "begin" "policy" string ":" string ":"
        "state-space" ":" "{" state_space "}" ";"
        "input-vector" ":" "{" input_vector "}" ";"
        "begin" "authorisation-scheme" ":"
            authorisation_scheme
        "end" "authorisation-scheme" ";"
        "begin" "initial-space" ":"
            inital_state
        "end" "initial-space" ";"
        "begin" "extension-vector" ":"
            extension_vector
        "end" "extension-vector" ";"
    "end" "policy" ";"
    ;

state_space =
    state_space "," string
    | string
    ;

input_vector =
    input_vector "," "2" "^" string
    | input_vector "," string
    | "2" "^" string
    | string
    ;

authorisation_scheme =
    authorisation_scheme command
    | command
    ;

command =
    string "(" parameter_list_nonempty ")" ":"
        "condition" ":" condition ";"
    | string "(" parameter_list_nonempty ")" ":"
        "begin" "body" ":"
            body
        "end" "body" ";"
    | string "(" parameter_list_nonempty ")" ":"
```

```
        "condition" ":" condition ";"
        "begin" "body" ":"
            body
        "end" "body" ";"
    ;

condition =
    comparable "==" comparable
    | comparable "!=" comparable
    | comparable "in" string_or_function
    | comparable "not" "in" string_or_function
    | "query" "(" value "," "[" string_list "]" ")"
    | string "(" string_or_value_list_nonempty ")"
    | "(" condition ")"
    | "not" condition
    | condition "and" condition
    | condition "or" condition
    | "forall" string "in" string_or_function ":" condition
    | "exists" string "in" string_or_function ":" condition
    | "forall" tuple "in" string ":" condition
    | "exists" tuple "in" string ":" condition
    ;


string_or_function =
    string
    | string "(" string_or_value_list_nonempty ")"
    ;

comparable =
    string_or_function
    | tuple
    | tuple_value
    | value
    ;

body =
    body string "(" string_or_value_list_nonempty ")" ";"
    | string "(" string_or_value_list_nonempty ")" ";"
    ;

inital_state =
    inital_state state
    | state
    ;

state =
    string "=" "{" "}" ";"
    | string "=" "{" value_list_nonempty "}" ";"
    | string "=" "{" tuple_value_list_nonempty "}" ";"
    | string "=" "{" mapping_value_list_nonempty "}" ";"
    | string "=" "{" mapping_value_set_list_nonempty "}" ";"
    ;

extension_vector =
    /* empty */
    | extension_vector state
    ;

string_or_value_list_nonempty =
    string_or_value_list_nonempty "," string_or_value
    | string_or_value
    ;

string_or_value =
    value
    | string
    ;

value_list_nonempty =
    value_list_nonempty "," value
    | value
    ;

tuple_value =
    "[" value_list_nonempty "]"
    ;

tuple_value_list_nonempty =
    tuple_value_list_nonempty "," tuple_value
    | tuple_value
    ;

mapping_value_list_nonempty =
    mapping_value_list_nonempty "," mapping_value
    | mapping_value
    ;

mapping_value =
    "(" value_list_nonempty ":" value ")"
    ;

mapping_value_set_list_nonempty =
    mapping_value_set_list_nonempty "," mapping_value_set
    | mapping_value_set
    ;

mapping_value_set =
    "(" value_list_nonempty ":" "{" value_list_nonempty "}" ")"
    ;
```

```
tuple =
    "[" string_list_nonempty "]"
    ;

value =
    "'" string "'"
    ;

string_list =
    /* empty */
    | string_list_nonempty
    ;

string_list_nonempty =
    string_list_nonempty "," string
    | string
    ;

parameter_list_nonempty =
    parameter_list_nonempty "," string string
    | parameter_list_nonempty "," "2" "^" string string
    | string string
    | "2" "^" string string
    ;

string =
    letter sub_string
    ;

sub_string =
    /* empty */
    | sub_string letter
    | sub_string digit
    | sub_string symbol
    ;

letter = "a" | ... | "z" | "A" | ... | "Z" ;

digit = "0" | ... | "9" ;

symbol = "-" | "_" ;
```