1 Modeling

We present the specification of a mutual exclusion protocol.

1.1 Rigid data types

Example 1 (Labels).

```
 spec! LABEL
 sort Label .
 ops re wt cs : -> Label [ctor].
 op _~_ : Label Label -> Bool [comm].
 var  L : Label .
 eq (re ~ wt) = false .
 eq (re ~ cs) = false .
 eq (wt ~ cs) = false .
---
 ceq true = false if re = wt .
 ceq true = false if re = cs .
 ceq true = false if wt = cs .
```

Example 2 (Process identifiers).

```
 spec* PID
 inc BOOL .
 sort Pid .
 op _~_ : Pid Pid -> Bool [comm].
 vars I J : Pid .
 eq I ~ I = true .
 ceq I = J if I ~ J [nonexec].
```

Example 3 (lists of process identifiers).

```
 spec! SEQUENCE{X :: PID}
 sort Sequence .
 subsorts X$Pid < Sequence .
--- constructors
 op empty : -> Sequence [ctor] .
 op _,_ : Sequence Sequence -> Sequence [ctor id: empty assoc].
 vars Q Q' : Sequence . var I : X$Pid .
---
 op top : Sequence -> X$Pid .
 eq top(empty) = empty .
 eq top(I,Q) = I .
---
 op get : Sequence -> Sequence .
 eq get(empty) = empty .
 eq get(I,Q) = Q .
---
 ceq true = false if Q,I,Q' := empty .
 ceq [lemma-top]: top(Q,I) = top(Q) if top(Q) :: X$Pid .
```

1.2 Nominals

Example 4 (Agents).

```
 spec* AGENT
 sort Agent
```

Example 5 (Nominals).

```
spec! NOMINAL{Y :: AGENT}
sorts Sys.
--- actions
op init : -> Sys [ctor].
ops want try exit : Sys Y$Agent -> Sys [ctor].
```

1.3 Flexible data types

Example 6 (Mutual exclusion protocol).

spec* QLOCK{X :: PID, Y :: AGENT}

pr SEQUENCE{X} . pr NOMINAL{Y} . pr LABEL .

--- observers

op pid $:\to$ X\$Pid --- extract pid from agents

op sq $:\to$ Sequence --- gives the waiting queue for each state

op pc $:\to$ Label --- indicates the label of each agent at a given state

--- variables

vars S $\mathtt{S}_1$ $\mathtt{S}_2$: Sys

vars I J K : X\$Pid

vars A B C : Y\$Agent

var Q : Sequence

--- restrictions ---

(1) $\forall \mathtt{A}, \mathtt{S}_1, \mathtt{S}_2 \cdot @_{\mathtt{S}_1} @_{\mathtt{A}} \mathsf{pid} = @_{\mathtt{S}_2} @_{\mathtt{A}} \mathsf{pid}$ --- pid depends only of the agent

(2) $\forall \mathtt{A}, \mathtt{B}, \mathtt{S} \cdot @_{\mathtt{S}} @_{\mathtt{A}} \mathsf{sq} = @_{\mathtt{S}} @_{\mathtt{B}} \mathsf{sq}$ --- sq depends only of the current state

--- init ---

(3) $\forall \mathtt{A} \cdot @_{\mathtt{init}} @_{\mathtt{A}} \mathsf{pc} = \mathtt{re}$

(4) $@_{\mathtt{init}} \mathsf{sq} = \mathsf{empty}$

--- want ---

(5) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{want(S,A)}} @_{\mathtt{B}} \mathsf{pc} = \mathtt{wt}$ if $@_{\mathtt{S}} @_{\mathtt{A}} \mathsf{pc} = \mathtt{re} \ \bigwedge \ \mathtt{A} = \mathtt{B}$

(6) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{want(S,A)}} @_{\mathtt{B}} \mathsf{pc} = @_{\mathtt{S}} @_{\mathtt{B}} \mathsf{pc}$ if $\neg(\mathtt{A} = \mathtt{B})$

(7) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{want(S,A)}} @_{\mathtt{B}} \mathsf{pc} = @_{\mathtt{S}} @_{\mathtt{B}} \mathsf{pc}$ if $\neg(@_{\mathtt{A}} @_{\mathtt{S}} \mathsf{pc} = \mathtt{re})$

(8) $\forall \mathtt{S}, \mathtt{A} \cdot @_{\mathtt{want(S,A)}} \mathsf{sq} = (@_{\mathtt{S}} \mathsf{sq}), (@_{\mathtt{A}} \mathsf{pid})$ if $@_{\mathtt{S}} @_{\mathtt{A}} \mathsf{pc} = \mathtt{re}$

(9) $\forall \mathtt{S}, \mathtt{A} \cdot @_{\mathtt{want(S,A)}} \mathsf{sq} = @_{\mathtt{S}} \mathsf{sq}$ if $\neg(@_{\mathtt{S}} @_{\mathtt{A}} \mathsf{pc} = \mathtt{re})$

--- try ---

(10) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{try(S,A)}} @_{\mathtt{B}} \mathsf{pc} = \mathtt{cs}$ if $@_{\mathtt{S}} @_{\mathtt{A}} \mathsf{pc} = \mathtt{wt} \ \bigwedge \ (@_{\mathtt{A}} \mathsf{pid}), \mathtt{Q} := @_{\mathtt{S}} \mathsf{sq} \ \bigwedge \ \mathtt{A} = \mathtt{B}$

(11) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{try(S,A)}} @_{\mathtt{B}} \mathsf{pc} = @_{\mathtt{S}} @_{\mathtt{B}} \mathsf{pc}$ if $\neg(\mathtt{A} = \mathtt{B})$

(12) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{try(S,A)}}\, @_{\mathtt{B}}\, \mathtt{pc} = @_{\mathtt{S}}\, @_{\mathtt{B}}\, \mathtt{pc}$ `if` $\neg(@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{wt})$

(13) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{B}}\, @_{\mathtt{try(S,A)}}\, \mathtt{pc} = @_{\mathtt{B}}\, @_{\mathtt{S}}\, \mathtt{pc}$ `if` $\neg(\mathtt{top}(@_{\mathtt{S}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid})$

(14) $\forall \mathtt{S}, \mathtt{A} \cdot @_{\mathtt{try(S,A)}}\, \mathtt{sq} = @_{\mathtt{S}}\, \mathtt{sq}$

`--- exit ---`

(15) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{exit(S,A)}}\, @_{\mathtt{B}}\, \mathtt{pc} = \mathtt{re}$ `if` $@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}\ \bigwedge\ \mathtt{A} = \mathtt{B}$

(16) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{exit(S,A)}}\, @_{\mathtt{B}}\, \mathtt{pc} = @_{\mathtt{A}}\, @_{\mathtt{B}}\, \mathtt{pc}$ `if` $\neg(\mathtt{A} = \mathtt{B})$

(17) $\forall \mathtt{S}, \mathtt{A}, \mathtt{B} \cdot @_{\mathtt{exit(S,A)}}\, @_{\mathtt{B}}\, \mathtt{pc} = @_{\mathtt{A}}\, @_{\mathtt{B}}\, \mathtt{pc}$ `if` $\neg(@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs})$

(18) $\forall \mathtt{S}, \mathtt{A} \cdot @_{\mathtt{exit(S,A)}}\, \mathtt{sq} = \mathtt{get}(@_{\mathtt{S}}\, \mathtt{sq})$ `if` $@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}$

(19) $\forall \mathtt{S}, \mathtt{A} \cdot @_{\mathtt{exit(S,A)}}\, \mathtt{sq} = @_{\mathtt{S}}\, \mathtt{sq}$ `if` $\neg(@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs})$

2 Formal verification

We are interested in proving formally $\mathtt{QLOCK} \vdash \forall \mathtt{S}, \mathtt{A} \cdot \mathtt{top}(@_{\mathtt{S}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid}$ if $@_{\mathtt{S}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}$.

==

spec $\mathtt{QLOCK}_I$

`pr QLOCK .`

`op s : -> Sys .`

(20) $\forall \mathtt{A} \cdot \mathtt{top}(@_{\mathtt{s}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid}$ `if` $@_{\mathtt{s}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}$ `[induction hypothesis].`

==

spec $\mathtt{QLOCK}_{TC}$

pr $\mathtt{QLOCK}_I$.

`op a b : -> Y$Agent .`

==

Apply induction on S:

[init] In this case,

1	$\mathtt{QLOCK} \vdash \forall \mathtt{A} \cdot \mathtt{top}(@_{\mathtt{init}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid}$ if $@_{\mathtt{init}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}$	
2	$\mathtt{QLOCK} \vdash \forall \mathtt{A} \cdot \mathtt{top}(@_{\mathtt{init}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid}$ if $\mathtt{re} = \mathtt{cs}$	by sentence (3)
3	discharged	since $\mathtt{QLOCK} \vdash \mathtt{true} = \mathtt{false}$ if $\mathtt{re} = \mathtt{cs}$

[want]

1	$\mathtt{QLOCK}_I \vdash \forall \mathtt{A}, \mathtt{B} \cdot \mathtt{top}(@_{\mathtt{want(s,B)}}\, \mathtt{sq}) = @_{\mathtt{A}}\, \mathtt{pid}$ if $@_{\mathtt{want(s,B)}}\, @_{\mathtt{A}}\, \mathtt{pc} = \mathtt{cs}$	
2	$\mathtt{QLOCK}_{TC} \vdash \mathtt{top}(@_{\mathtt{want(s,b)}}\, \mathtt{sq}) = @_{\mathtt{a}}\, \mathtt{pid}$ if $@_{\mathtt{want(s,b)}}\, @_{\mathtt{a}}\, \mathtt{pc} = \mathtt{cs}$	
2.1	$\mathtt{QLOCK}_{TC} + \{\mathtt{b} = \mathtt{a}, @_{\mathtt{s}}\, @_{\mathtt{b}}\, \mathtt{pc} = \mathtt{re}\} \vdash$ $\mathtt{top}(@_{\mathtt{want(s,b)}}\, \mathtt{sq}) = @_{\mathtt{a}}\, \mathtt{pid}$ if $@_{\mathtt{want(s,b)}}\, @_{\mathtt{a}}\, \mathtt{pc} = \mathtt{cs}$	by case analysis
2.2	$\mathtt{QLOCK}_{TC} + \{\mathtt{a} = \mathtt{b}, @_{\mathtt{s}}\, @_{\mathtt{b}}\, \mathtt{pc} = \mathtt{re}\} \vdash$ $\mathtt{top}(@_{\mathtt{s}}\, \mathtt{sq}, @_{\mathtt{b}}\, \mathtt{pid}) = @_{\mathtt{b}}\, \mathtt{pid}$ if $\mathtt{re} = \mathtt{cs}$	by rew
2.3	discharged	since $\mathtt{QLOCK} \vdash \mathtt{true} = \mathtt{false}$ if $\mathtt{re} = \mathtt{cs}$

3

[try]

[exit]