

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:→ X$Pid --- extract pid from agents
op sq:→ Sequence --- gives the waiting queue for each state
op pc:→ Label --- indicates the label of each agent at a given state

--- variables

vars S S1 S2 : Sys
vars I J K : X$Pid
vars A B C : Y$Agent
var Q : Sequence

--- restrictions ---

(1)  $\forall A, S_1, S_2. @_{S_1} @_A \text{pid} = @_{S_2} @_A \text{pid}$  --- pid depends only of the agent
(2)  $\forall A, B, S. @_S @_A \text{sq} = @_S @_B \text{sq}$  --- sq depends only of the current state

--- init ---

(3)  $\forall A. @_{\text{init}} @_A \text{pc} = \text{re}$ 
(4)  $@_{\text{init}} \text{sq} = \text{empty}$ 

--- want ---

(5)  $\forall S, A, B. @_{\text{want}(S,A)} @_B \text{pc} = \text{wt}$  if  $@_S @_A \text{pc} = \text{re} \wedge A = B$ 
(6)  $\forall S, A, B. @_{\text{want}(S,A)} @_B \text{pc} = @_S @_B \text{pc}$  if  $\neg(A = B)$ 
(7)  $\forall S, A, B. @_{\text{want}(S,A)} @_B \text{pc} = @_S @_B \text{pc}$  if  $\neg(@_A @_S \text{pc} = \text{re})$ 
(8)  $\forall S, A. @_{\text{want}(S,A)} \text{sq} = (@_S \text{sq}), (@_A \text{pid})$  if  $@_S @_A \text{pc} = \text{re}$ 
(9)  $\forall S, A. @_{\text{want}(S,A)} \text{sq} = @_S \text{sq}$  if  $\neg(@_S @_A \text{pc} = \text{re})$ 

--- try ---

(10)  $\forall S, A, B. @_{\text{try}(S,A)} @_B \text{pc} = \text{cs}$  if  $@_S @_A \text{pc} = \text{wt} \wedge (@_A \text{pid}), Q := @_S \text{sq} \wedge A = B$ 
(11)  $\forall S, A, B. @_{\text{try}(S,A)} @_B \text{pc} = @_S @_B \text{pc}$  if  $\neg(A = B)$ 
```

- (12) $\forall S, A, B \cdot @_{\text{try}(S,A)} @_B \text{pc} = @_S @_B \text{pc} \text{ if } \neg(@_S @_A \text{pc} = \text{wt})$
- (13) $\forall S, A, B \cdot @_B @_{\text{try}(S,A)} \text{pc} = @_B @_S \text{pc} \text{ if } \neg(\text{top}(@_S \text{sq}) = @_A \text{pid})$
- (14) $\forall S, A \cdot @_{\text{try}(S,A)} \text{sq} = @_S \text{sq}$
- exit ---
- (15) $\forall S, A, B \cdot @_{\text{exit}(S,A)} @_B \text{pc} = \text{re} \text{ if } @_S @_A \text{pc} = \text{cs} \wedge A = B$
- (16) $\forall S, A, B \cdot @_{\text{exit}(S,A)} @_B \text{pc} = @_A @_B \text{pc} \text{ if } \neg(A = B)$
- (17) $\forall S, A, B \cdot @_{\text{exit}(S,A)} @_B \text{pc} = @_A @_B \text{pc} \text{ if } \neg(@_S @_A \text{pc} = \text{cs})$
- (18) $\forall S, A \cdot @_{\text{exit}(S,A)} \text{sq} = \text{get}(@_S \text{sq}) \text{ if } @_S @_A \text{pc} = \text{cs}$
- (19) $\forall S, A \cdot @_{\text{exit}(S,A)} \text{sq} = @_S \text{sq} \text{ if } \neg(@_S @_A \text{pc} = \text{cs})$

2 Formal verification

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

=====

spec QLOCK_I

pr QLOCK .

op $s : \rightarrow \text{Sys}$.

- (20) $\forall A \cdot \text{top}(@_S \text{sq}) = @_A \text{pid} \text{ if } @_S @_A \text{pc} = \text{cs} \text{ [induction hypothesis]}.$

=====

spec QLOCK_{TC}

pr QLOCK_I .

op $a \ b : \rightarrow \text{Y\$Agent}$.

=====

Apply induction on S:

[init] In this case,

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

[want]

- | | | |
|-----|---|--|
| 1 | $\text{QLOCK}_I \vdash \forall A, B \cdot \text{top}(@_{\text{want}(S,B)} \text{sq}) = @_A \text{pid} \text{ if } @_{\text{want}(S,B)} @_A \text{pc} = \text{cs}$ | |
| 2 | $\text{QLOCK}_{TC} \vdash \text{top}(@_{\text{want}(S,b)} \text{sq}) = @_a \text{pid} \text{ if } @_{\text{want}(S,b)} @_a \text{pc} = \text{cs}$ | |
| 2.1 | $\text{QLOCK}_{TC} + \{b = a, @_S @_b \text{pc} = \text{re}\} \vdash$ | by case analysis |
| | $\text{top}(@_{\text{want}(S,b)} \text{sq}) = @_a \text{pid} \text{ if } @_{\text{want}(S,b)} @_a \text{pc} = \text{cs}$ | |
| 2.2 | $\text{QLOCK}_{TC} + \{a = b, @_S @_b \text{pc} = \text{re}\} \vdash$ | by rew |
| | $\text{top}(@_S \text{sq}, @_b \text{pid}) = @_b \text{pid} \text{ if } \text{wt} = \text{cs}$ | |
| 2.3 | discharged | since $\text{QLOCK} \vdash \text{true} = \text{false} \text{ if } \text{wt} = \text{cs}$ |

[try]

[exit]