



# Chapter 6

## Model checking for propositional modal logic

To automate verification of modal formulas against a system, we use the semantics of the system, the model  $M$ , to check the formula against.

A model check algorithm should be able to decide whether or not a formula is true in a state of the model, in particular whether or not a formula is true in the starting state of a model. For modal formulas, being true in a state does not only depend on that state, but can also involve other states and transitions from them. Because these models can be large and may involve transitions that form loops it is not possible to see immediately, in one step, whether or not a formula is true in a state.

A model checking algorithm is a recipe that enables to decide the truth of a formula over a system by stepwise operations that each involve not more than inspection of a state and its outgoing transitions and the states attached there: it is, e.g., not allowed to inspect sequences of states.

### 6.1 The model checking algorithm

The idea for modal model checking is to stepwise label the states of the model with sub formulas of the formula to be checked, building-up from the smallest ones, propositions, to the original formula. At each step, a formula should be added as a label to just those states at which it is true.

Any modal formula can be rewritten using only the symbols  $\wedge$ ,  $\neg$  and  $\Box$ , a so-called *adequate* set of symbols for modal logic. We assume that formulas are of that form and provide the algorithm for such formulas.

Table 6.1: Table for Mutual Exclusion for single  $t_1 t_2$  state

	$w_0$	$w_1$	$w_2$	$w_3$	$w_4$	$w_5$	$w_6$	$w_7$
$c_1$					✓		✓	
$c_2$						✓		✓
$c_1 \wedge c_2$								
$\neg(c_1 \wedge c_2)$	✓	✓	✓	✓	✓	✓	✓	✓

**Definition 6.1.1** *Suppose  $\psi$  is a sub formula of the formula to be checked and that all the immediate sub formulas of  $\psi$  have been checked.*

*For each state  $w$  in the model, if  $\psi$  is*

- $p$ , label  $w$  with  $\psi$  if  $V(s)(p) = tt$ ;
- $\neg\varphi$ , label  $w$  with  $\psi$  if  $w$  was not labeled with  $\varphi$ ;
- $\varphi_1 \wedge \varphi_2$ , label  $w$  with  $\psi$  if  $w$  was labeled with  $\varphi_1$  and  $\varphi_2$ ;
- $\Box\varphi$ , label  $w$  with  $\psi$  if for all outgoing edges and states  $w \rightarrow w'$ ,  $w'$  was labeled with  $\varphi$ .

This algorithm terminates, as the size of the sub formulas increases. At termination, the states at which the original formula is added as a label are the states at which the original formula holds.

Often it is possible, and when checking by hand easier, to extend the model check algorithm itself rather than to translate the formula into adequate symbols form, e.g., to extend the algorithm with a clause for  $\Diamond\varphi$  rather than use  $\neg\Box\neg\varphi$ .

We apply the algorithm to Mutual Exclusion and Eventual Access.

- Mutual Exclusion:  $M \models \neg(c_1 \wedge c_2)$ .
- Eventual Access:  $M \models t_1 \Rightarrow \Box(c_1 \vee \Box(c_1 \vee \Box(c_1 \vee \Box c_1)))$  - "If  $t_1$  then in at most four steps  $c_1$ ".

# Chapter 7

## Model checking for CTL

The idea of model checking for *CTL* is essentially the same as for modal logic, stepwise labeling of states.

### 7.1 The model checking algorithm

Any *CTL* formula can be rewritten using only the symbols  $\wedge, \neg, EX, AF, E(\dots U \dots)$ , an adequate set of symbols for modal logic. We assume that formulas are of that form and provide the algorithm for such formulas.

**Definition 7.1.1** *Suppose  $\psi$  is a sub formula of the formula to be checked and that all the immediate sub formulas of  $\psi$  have been checked.*

*For each state  $w$  in the model, if  $\psi$  is*

- $p$ , label  $w$  with  $\psi$  if  $V(s)(p) = tt$ ;
- $\neg\varphi$ , label  $w$  with  $\psi$  if  $w$  was not labeled with  $\varphi$ ;
- $\varphi_1 \wedge \varphi_2$ , label  $w$  with  $\psi$  if  $w$  was labeled with  $\varphi_1$  and  $\varphi_2$ ;
- $EX\varphi$ , label  $w$  with  $\psi$  if at least one of its successors was labeled with  $\varphi$ ;
- $AF\varphi$ ,
  - label  $w$  with  $\psi$  if  $w$  was labeled with  $\varphi$ ,
  - repeat: label  $w$  with  $\psi$  if all successors were labeled with  $\psi$ , until there is no change;
- $E(\varphi_1 U \varphi_2)$ ,

Table 7.1: Table for Eventual Access for  $t_1t_2l$  and  $t_1t_2r$  states

	$w_0$	$w_1$	$w_2$	$w_3l$	$w_3r$	$w_4$	$w_5$	$w_6$	$w_7$
$t_1$		✓		✓	✓				✓
$\neg t_1$	✓		✓			✓	✓	✓	
$c_1$						✓		✓	
$AFc_1S$						✓		✓	
$AFc_1R$				✓		✓		✓	
$AFc_1 - R$		✓		✓		✓		✓	
$AFc_1 - R$		✓		✓		✓		✓	✓
$AFc_1 - R$		✓		✓	✓	✓		✓	✓
$AFc_1 - T$		✓		✓	✓	✓		✓	✓
$\neg t_1 \vee AFc_1$	✓	✓	✓	✓	✓	✓	✓	✓	✓

- label  $w$  with  $\psi$  if  $w$  was labeled with  $\varphi_2$ ,
- repeat: label  $w$  with  $\psi$  if  $w$  was labeled with  $\varphi_1$  and at least one successor was labeled with  $\psi$ , until there is no change.

This algorithm terminates, as the size of the sub formulas increases. At termination, the states at which the original formula is added as a label are the states at which the original formula holds.

Often it is possible, and when checking by hand easier, to extend the model check algorithm itself rather than to translate the formula into adequate symbols form, e.g., to extend the algorithm with a clause for  $EF\varphi$  rather than use  $E(\mathbf{true}U\varphi_2)$ ; similar for  $AX$ ,  $EG$ ,  $AG$ ,  $A(\dots U \dots)$ .

We apply the algorithm to Mutual Exclusion and Eventual Access and Non-blocking.

- Mutual Exclusion:  $M \models AG\neg(c_1 \wedge c_2)$ .
- Eventual Access:  $M \models AG(t_1 \Rightarrow AFc_1)$ .
- Non-blocking:  $M \models AG(n_1 \Rightarrow Eft_1)$ .