July 6, 2021

1 The Syntax of Computation Tree Logic

Computation tree logic (CTL) was introduced by Turing award winners Clarke and Emerson [3]. The formulas of this logic consist of the constants `true` and `false` and so-called atomic propositions which are combined by means of several operators that we will discuss below. The *atomic propositions* are used to express basic facts about the states of the system. That is, these atomic propositions are state predicates. In the next section, we provide some concrete examples of atomic propositions in the context of Java code.

CTL contains the operators

- negation, denoted $\neg$,
- conjunction, denoted by $\wedge$,
- disjunction, denoted $\vee$,
- implication, denoted $\rightarrow$, and
- equivalence, denoted $\leftrightarrow$.

Furthermore, it contains

- universal quantification, denoted $\forall$, and
- existential quantification, denoted $\exists$.

Finally, it contains the so-called temporal operators

- next, denoted $\bigcirc$,
- until, denoted U,
- always, denoted $\Box$, and
- eventually, denoted $\Diamond$.

Let us formally define the syntax of CTL. Let AP be the set of atomic propositions. The set of CTL formulas is defined by the following grammar.

$$\begin{aligned}\varphi ::= & \,(\varphi) \mid a \\ & \mid \texttt{true} \mid \texttt{false} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \\ & \mid \forall \bigcirc \varphi \mid \exists \bigcirc \varphi \mid \forall \varphi \,\mathsf{U}\, \varphi \mid \exists \varphi \,\mathsf{U}\, \varphi \mid \forall \Box \varphi \mid \exists \Box \varphi \mid \forall \Diamond \varphi \mid \exists \Diamond \varphi\end{aligned}$$

where $a \in AP$.

In order to make sense of a CTL formula such as

$$\forall \bigcirc a \rightarrow b \rightarrow c$$

we need to define the precedence of the operators. Furthermore, we need to specify whether the binary operators are left or right associative. For the order of precedence, we use the commonly accepted order (from highest to lowest): $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. According to Baier and Katoen [1], U takes precedence over $\wedge$, $\vee$, and $\rightarrow$ (they do not consider $\leftrightarrow$). Usually, unary operators have higher precedence than binary ones. Hence, the operators, listed from highest to lowest precedence, are

$$\begin{array}{l}\neg \\ \forall\bigcirc, \exists\bigcirc, \forall\Box, \exists\Box, \forall\Diamond, \exists\Diamond \\ \forall\mathsf{U}, \exists\mathsf{U} \\ \wedge \\ \vee \\ \rightarrow \\ \leftrightarrow\end{array}$$

The binary operators $\wedge$, $\vee$ and $\leftrightarrow$ are (left) associative. Usually, $\rightarrow$ is considered right associative. According to Baier and Katoen [1], U is also right associative.

Using the above specified precedence and associativity rules, the above CTL formula is interpreted as

$$(\forall \bigcirc a) \rightarrow (b \rightarrow c)$$

To express the CTL formulas in ASCII, we use the following grammar.

$$\begin{aligned}\varphi ::= & \,(\varphi) \mid a \\ & \mid \texttt{true} \mid \texttt{false} \mid \texttt{!}\varphi \mid \varphi \,\texttt{\&\&}\, \varphi \mid \varphi \,\texttt{||}\, \varphi \mid \varphi \,\texttt{->}\, \varphi \mid \varphi \,\texttt{<->}\, \varphi \\ & \mid \texttt{AX}\,\varphi \mid \texttt{EX}\,\varphi \mid \varphi \,\texttt{AU}\, \varphi \mid \varphi \,\texttt{EU}\, \varphi \mid \texttt{AG}\,\varphi \mid \texttt{EG}\,\varphi \mid \texttt{AF}\,\varphi \mid \texttt{EF}\,\varphi\end{aligned}$$

The ASCII representation of $\neg$, $\wedge$, and $\vee$ is taken from Java. It is common practice to use `A` and `E` for universal (for *a*ll) and existential (*e*xists) quantification. In the seminal paper by Turing award winner Pnueli [7], the temporal operators $\bigcirc$, U, $\Box$, and $\Diamond$ are represented as `X` (ne*x*t), `U` (*u*ntil), `G` (*g*lobally), and `F` (*f*uture). The representation of $\forall\varphi \,\mathsf{U}\, \varphi$ as $\varphi \,\texttt{AU}\, \varphi$ is new, as far as we know. The above CTL formula is represented in ASCII as follows.

```
AX a -> b -> c
```

2 The Syntax of Computation Tree Logic for Java

The next operator $\bigcirc$ expresses that something holds in the next state. For Java code, if one were to define the notion of next state, it would probably be the state after the next bytecode instruction has been executed. However, expressing properties of Java code in terms to steps taken at the bytecode level seems of limited, if any, use. Therefore, we do not consider the next operator $\bigcirc$.

Recall that atomic propositions are used to express basic facts about the states. For now, we restrict our attention to static Boolean fields. Such an atomic proposition holds in those states in which the field has the value true. In Java, static Boolean fields are of the form

- $\langle$package name$\rangle$. $\langle$class name$\rangle$. $\langle$field name$\rangle$ or
- $\langle$class name$\rangle$. $\langle$field name$\rangle$.

For example, the package `java.awt` contains the classes `AWTEvent` and `InvocationEvent`. The former contains the static field `consumed` and the latter contains `catchExceptions`. Hence, the static Boolean field `java.awt.AWTEvent.consumed` is an atomic proposition, as is `java.awt.InvocationEvent.catchExceptions`. Those fields are used as atomic propositions in the following CTL formula.

```
AG (java.awt.AWTEvent.consumed
    || EF !java.awt.event.InvocationEvent.catchExceptions)
```

3 A Lexer and Parser for CTL Formulas

A lexer and parser for CTL formulas have been developed using ANTLR [6]. The above described grammar can be specified in ANTLR format as folows.

```
formula
  : '(' formula ')'                        #Bracket
  | ATOMIC_PROPOSITION                     #AtomicProposition
  | 'true'                                 #True
  | 'false'                                #False
  | '!' formula                            #Not
  | 'AG' formula                           #ForAllAlways
  | 'AF' formula                           #ForAllEventually
  | 'EG' formula                           #ExistsAlways
  | 'EF' formula                           #ExistsEventually
  | <assoc=right> formula 'AU' formula     #ForAllUntil
  | <assoc=right> formula 'EU' formula     #ExistsUntil
  | <assoc=left> formula '&&' formula      #And
  | <assoc=left> formula '||' formula      #Or
  | <assoc=right> formula '->' formula     #Implies
  | <assoc=left> formula '<->' formula     #Iff
```

The operators AU, EU, and -> are specified as right associative. The other binary operators are left associative. The second column of the above rule contains the labels of the alternatives (see [6, Section 8.2]). We will discuss their role below.

The order of the alternatives is consistent with the precedence of the operators (if an operator has higher precedence, then its alternative occurs earlier). As a consequence, we had to order the operators AU and EU. We gave AU higher precedence than EU. Assume that a, b, and c are atomic propositions. The formula a AU b AU c is equivalent to a AU (b AU c) since AU is right associative. The formula a AU b EU c is equivalent to (a AU b) EU c since AU binds stronger than EU. For the same reason, the formula a EU b AU c is equivalent to a AU (b EU c).

Recall that the atomic propositions are static attributes. To specify these, we used relevant snippets of the ANTLR grammar for Java[1] Whitespace, that is, spaces, tabs, form feeds, and returns are skipped.

Later, we add here a discussion of error handling.

4 From Parse Tree to Abstract Syntax Tree

Next, we translate a parse tree, generated by the lexer and parser, to an abstract syntax tree. An abstract syntax tree for CTL is represented by an object of type Formula, which is part of the package ctl. A UML diagram with the classes of the ctl package can be found in Figure 1. The CTL formula

```
AG (java.awt.AWTEvent.consumed
    || EF !java.awt.event.InvocationEvent.catchExceptions)
```

is represented by the following Formula object.

```
Formula formula =
  new ForAllAlways(
    new Or(
      new AtomicProposition("java.awt.AWTEvent.consumed"),
      new ExistsEventually(
        new Not(
          new AtomicProposition("java.awt.event.InvocationEvent.catchExcept
        )
      )
    )
  );
```

To implement this translation, we use the visitor design pattern. ANTLR supports this design pattern (see [6, Section 7.3]). From the CTL grammar, ANTLR generates a CTLVisitor interface. This interface contains a visit method for each alternative. For example, for the alternative labelled ExistsAlways, the interface contains the method visitExistsAlways.

[1] See github.com/antlr/grammars-v4/tree/master/java/java8.

ANTLR also generates the `CTLBaseVisitor` class. This adapter class provides a default implementation for all the methods of the `CTLVisitor` interface. We implement our translation by extending this class and overriding methods. For example, when we visit a node of the parse tree corresponding to the alternative labelled `And`, we first visit the left child and obtain the `Formula` object corresponding to the translation of the parse tree rooted at that left child. Next, we visit the right child and obtain the `Formula` object for the parse tree rooted at that right child. Finally, we create an `And` object from those two `Formula` objects.

```
@Override
public Formula visitAnd(AndContext context) {
  Formula left = (Formula) visit(context.formula(0));
  Formula right = (Formula) visit(context.formula(1));
  return new And(left, right);
}
```

Since the implication operator is right associative, in the `visitImplies` method we visit the right child first.

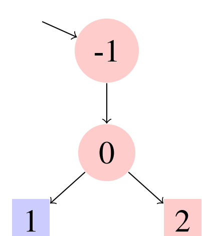
```
@Override
public Formula visitImplies(ImpliesContext context) {
  Formula right = (Formula) visit(context.formula(1));
  Formula left = (Formula) visit(context.formula(0));
  return new Implies(left, right);
}
```

5 Testing the Lexer, the Parser, and the Translation

6 A New Semantics for CTL

The normal semantics of CTL is described in [1, Section 6.2.2]. This normal semantics is defined for a transition system $(S, Act, \rightarrow, I, AP, L)$. Such a transition system is defined in [1, Definition 2.1]. The new semantics considers a partial transition system. A partial transition system is a tuple $(S, F, Act, \rightarrow, I, AP, L)$, where all components are defined as before and $F \subseteq S$ is a set of fully explored states. A transition system is called partial because the states $S \setminus F$ are not fully explored yet, that is, these states *have* transitions that are not part of the transition relation $\rightarrow$.

Consider the following partial transition system.



State -1 is the initial state. The states -1 and 0 are fully explored, and states 1 and 2 are not fully explored. Consider, for example, the CTL formula $\exists\Diamond$ blue. This formula holds in the above partial transition system, since state 1 is blue and can be reached from the initial state. The CTL formula $\forall\Box$ red does not hold for the same reason. Now consider the CTL formula $\forall\Box$ (red $\vee$ blue). The above partial transition system does not provide a counterexample to this formula as all states that can be reached from the initial state are either red or blue. However, since states 1 and 2 are not fully explored, either state may have a successor that is neither red nor blue. So, the best we can say is "don't know." Hence, whether a partial transition system satisfies a CTL formula can be answered as either yes ($\top$), no ($\bot$), or don't know (?).

Recall that the satisfaction relation $\models$, defined in [1, Definition 6.4], can be viewed as mapping a state s of a transition system and a CTL formula φ to a Boolean, that is, (s, φ) is mapped to true if $s \models \varphi$ and mapped to false otherwise. The satisfaction relation $\models$ for CTL formulas on partial transition systems can be viewed as a mapping from states and formulas to $\top$, $\bot$, and ?.

We modify the definition of a transition system, as given in [1, Definition 2.1], as follows.

Definition 1. A *partial transition system* is a tuple $(S, F, Act, \rightarrow, I, AP, L)$ consisting of

- a set S of *states*,
- a set $F \subseteq S$ of *fully explored states*,
- a set Act of *actions*,
- a transition relation $\rightarrow \subseteq S \times Act \times S$,
- a set $I \subseteq S$ of *initial states*,
- a set AP of *atomic propositions*, and
- a *labelling function* $L : S \rightarrow 2^{AP}$.

The difference between a partial transition system and an ordinary transition system is the set F of fully explored states. Since the set Act of actions does not play in the remainder, we will drop it from the definition and simplify the transition relation to $\rightarrow \subseteq S \times S$. The partial transition system depicted above can be formally defined as $(S, F, \rightarrow, I, AP, L)$ where

- $S = \{-1, 0, 1, 2\}$,
- $F = \{-1, 0\}$,
- $\rightarrow = \{(-1, 0), (0, 1), (0, 2)\}$,
- $I = \{-1\}$,
- $AP = \{\text{blue}, \text{red}\}$, and

- and the function $L : S \to 2^{AP}$ is defined by

$$\begin{aligned} L(-1) &= \{\text{red}\} \\ L(0) &= \{\text{red}\} \\ L(1) &= \{\text{blue}\} \\ L(2) &= \{\text{red}\} \end{aligned}$$

Due to the presence of unexplored states, we also revisit the definition of paths. We fix a set $\mathcal{S}$ and we assume that for each partial transition system we have that $S \subseteq \mathcal{S}$. We denote the set of nonempty and finite sequences of states in $\mathcal{S}$ by $\mathcal{S}^*$, the set of infinite sequences of states in $\mathcal{S}$ by $\mathcal{S}^\omega$, and the set of nonempty finite and infinite sequences of states in $\mathcal{S}$ by $\mathcal{S}^\infty$.

Definition 2. Let $(S, F, \to, I, AP, L)$ be a partial transition system.

- The nonempty and finite sequence $s_0 s_1 \ldots s_n$ in $\mathcal{S}^*$, where $n \geq 0$, is a path if $s_i \to s_{i+1}$ for all $0 \leq i < n$ and $s_n \not\to$ and $s_n \in F$.
- The infinite sequence $s_0 s_1 \ldots$ in $\mathcal{S}^\omega$ is a path if $s_i \to s_{i+1}$ for all $i \geq 0$.

Note that we require that the final state of a finite path is fully explored.

Definition 3. Let $(S, F, \to, I, AP, L)$ be a partial transition system.

- The nonempty and finite sequence $s_0 s_1 \ldots s_n$ in $\mathcal{S}^*$, where $n \geq 0$, is a partial path if $s_i \to s_{i+1}$ for all $0 \leq i < n$.

We denote the set of (actual) paths defined in Definition 2 and 3 that start in state s by $APaths(s)$.

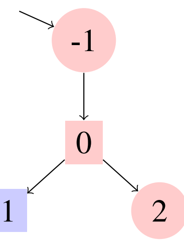
Definition 4. Let $(S, F, \to, I, AP, L)$ be a partial transition system.

- The nonempty and finite sequence $s_0 s_1 \ldots s_n s_{n+1} \ldots s_{n+m}$ in $\mathcal{S}^*$, where $n \geq 0$ and $m \geq 1$, is a potential path if $s_i \to s_{i+1}$ for all $0 \leq i < n$ and $s_n \notin F$ and $s_n \not\to s_{n+1}$.
- The infinite sequence $s_0 s_1 \ldots s_n s_{n+1} \ldots$ in $\mathcal{S}^\omega$, where $n \geq 0$, is a potential path if $s_i \to s_{i+1}$ for all $0 \leq i < n$ and $s_n \notin F$ and $s_n \not\to s_{n+1}$.

In the first case, the sequence $s_0 s_1 \ldots s_n s_{n+1} \ldots s_{n+m}$ consists of two parts: $s_0 s_1 \ldots s_n$ traverses the explored part of the partial transition system, whereas $s_{n+1} \ldots s_{n+m}$ traverses the unexplored part. Note that we require that $s_n \not\to s_{n+1}$: otherwise s_{n+1} would belong to the explored part. Similarly, the sequence $s_0 s_1 \ldots s_n s_{n+1} \ldots$ consists of the parts $s_0 s_1 \ldots s_n$ and $s_{n+1} \ldots$.

We denote the set of potential paths that start in state s by $PPaths(s)$. We denote the set of all paths that start in state s by $Paths(s)$.

Consider the following partial transition system.



We have that

$$APaths(-1) = \{-1, -1\ 0, -1\ 0\ 1, -1\ 0\ 2\}$$
$$PPaths(-1) = \{\, -1\ 0\ \pi \mid \pi[0] \not\in \{1, 2\} \wedge \pi \in S^\infty \,\} \cup \{\, -1\ 0\ 1\ \pi \mid \pi \in S^\infty \,\}$$

Since state 0 is not fully explored yet, we know that this state has more outgoing transitions than the two depicted in the above diagram. All the potential paths starting with $-1\ 0$ do not start with either $-1\ 0\ 1$ or $-1\ 0\ 2$. Since state 1 is not fully explored, it is not a final state and, as a result, the sequence $-1\ 0\ 1$ is *not* a potential path.

A partial transition system in which all states are fully explored, that is, an ordinary transition system, has no potential paths.

Proposition 1. *If $F = S$ then $PPAths(s) = \emptyset$ for all $s \in S$.*

Proof. Immediate from the definition of potential paths. □

The satisfaction relation $\models$ for CTL for ordinary transition systems is defined in [1, Definition 6.4]. It can be viewed as a function $[\![\ ]\!]$ that maps each CTL formula and state of a transition system to either true of false, that is, for each state formula φ and state s,

$$s \models \varphi \text{ iff } [\![\varphi]\!](s) = \text{true}.$$

To deal with partial transition systems, we extend the range of the function $[\![\ ]\!]$. Given a formula φ and a state s, we have that either

- $[\![\varphi]\!](s) = \top$: the formula φ holds in the state s,
- $[\![\varphi]\!](s) = \perp$: the formula φ does not hold in the state s, or
- $[\![\varphi]\!](s) = ?$: we cannot determine whether the formula φ holds in the state s since some states, relevant to φ, have not been explored.

For example, consider the partial transition system depicted above. Consider the formula $\forall \bigcirc$ red. We have that $[\![\forall \bigcirc \text{red}]\!](-1) = \top$ since the state -1 is fully explored and all its successor states are red. Furthermore, $[\![\forall \bigcirc \text{red}]\!](0) = \perp$ since one of the successor states of state 0 is not red. Finally, $[\![\forall \bigcirc \text{red}]\!](1) = ?$ since the state 1 is not fully explored and it does not have a successor that is not red.

We denote the set of three values, $\top$, $\perp$ and $?$ by $\mathbb{V}$, that is,

$$\mathbb{V} = \{\top, \perp, ?\}.$$

We can extend the usual Boolean operators to $\mathbb{V}$ as follows. Negation is captured in the following table.

v	$\neg v$
$\top$	$\bot$
$\bot$	$\top$
$?$	$?$

Conjunction is defined as follows.

$v \wedge w$		w		
		$\top$	$\bot$	$?$
	$\top$	$\top$	$\bot$	$?$
v	$\bot$	$\bot$	$\bot$	$\bot$
	$?$	$?$	$\bot$	$?$

Disjunction is defined as follows.

$v \vee w$		w		
		$\top$	$\bot$	$?$
	$\top$	$\top$	$\top$	$\top$
v	$\bot$	$\top$	$\bot$	$?$
	$?$	$\top$	$?$	$?$

Implication is defined as follows.

$v \rightarrow w$		w		
		$\top$	$\bot$	$?$
	$\top$	$\top$	$\bot$	$?$
v	$\bot$	$\top$	$\top$	$\top$
	$?$	$\top$	$\bot$	$?$

Equivalence is defined as follows.

$v \leftrightarrow w$		w		
		$\top$	$\bot$	$?$
	$\top$	$\top$	$\bot$	$?$
v	$\bot$	$\bot$	$\top$	$?$
	$?$	$?$	$\bot$	$?$

We denote the length of a (potential) path π by $|\pi|$. If the (potential) path π is infinite, then $|\pi| = \omega$. We denote the set of CTL formulas by CTL.

Definition 5. Let $(S, F, \rightarrow, I, AP, L)$ be a partial transition system. The function

$$[\![\ ]\!] : CTL \rightarrow S \rightarrow \mathbb{V}$$

is defined by structural induction on the CTL formula as follows.

- $[\![a]\!](s) = \begin{cases} \top & \text{if } a \in L(s) \\ \bot & \text{otherwise} \end{cases}$

- $[\![\texttt{true}]\!](s) = \top$
- $[\![\texttt{false}]\!](s) = \bot$
- $[\![\neg\varphi]\!](s) = \neg[\![\varphi]\!](s)$
- $[\![\varphi \wedge \psi]\!](s) = [\![\varphi]\!](s) \wedge [\![\psi]\!](s)$
- $[\![\varphi \vee \psi]\!](s) = [\![\varphi]\!](s) \vee [\![\psi]\!](s)$
- $[\![\varphi \rightarrow \psi]\!](s) = [\![\varphi]\!](s) \rightarrow [\![\psi]\!](s)$
- $[\![\varphi \leftrightarrow \psi]\!](s) = [\![\varphi]\!](s) \leftrightarrow [\![\psi]\!](s)$
- $[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[1]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \\ \bot & \text{if } [\![\varphi]\!](\pi[1]) = \bot \text{ for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1 \\ ? & \text{otherwise} \end{cases}$
- $[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[1]) = \top \text{ for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1 \\ \bot & \text{if } [\![\varphi]\!](\pi[1]) = \bot \text{ for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \Box \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \\ \bot & \text{if } [\![\varphi]\!](\pi[i]) = \bot \text{ for some } \pi \in \mathit{APaths}(s) \text{ and } 0 \leq i < |\pi| \\ ? & \text{otherwise} \end{cases}$
- $[\![\exists \Box \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[i]) = \top \text{ for some } \pi \in \mathit{APaths}(s) \text{ and for all } 0 \leq i < |\pi| \\ \bot & \text{if } [\![\varphi]\!](\pi[i]) = \bot \text{ for all } \pi \in \mathit{Paths}(s) \text{ and for some } 0 \leq i < |\pi| \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \Diamond \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and for some } 0 \leq i < |\pi| \\ \bot & \text{if } [\![\varphi]\!](\pi[i]) = \bot \text{ for some } \pi \in \mathit{APaths}(s) \text{ and for all } 0 \leq i < |\pi| \\ ? & \text{otherwise} \end{cases}$
- $[\![\exists \Diamond \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](\pi[i]) = \top \text{ for some } \pi \in \mathit{APaths}(s) \text{ and } 0 \leq i < |\pi| \\ \bot & \text{if } [\![\varphi]\!](\pi[i]) = \bot \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \varphi \,\mathsf{U}\, \psi]\!](s) =$ details still need to be added here.
- $[\![\exists \varphi \,\mathsf{U}\, \psi]\!](s) =$ details still need to be added here.

For a partial transition system in which all states are fully explored, that is, an ordinary transition system, $[\![\ ]\!]$ corresponds to $\models$ as defined in [1, Definition 6.4] adjusted to deal with finite paths as follows:

- $s \models \forall \bigcirc \varphi$ iff for all $\pi \in \mathit{APaths}(s)$ with $|\pi| > 1$, $\pi[1] \models \varphi$

- $s \models \exists \bigcirc \varphi$ iff for some $\pi \in APaths(s)$ with $|\pi| > 1$, $\pi[1] \models \varphi$
- $s \models \forall \varphi \mathsf{U} \psi$ iff for all $\pi \in APaths(s)$, there exists $0 \leq j < |\pi|$ such that $\pi[j] \models \psi$ and for all $0 \leq k < j$, $\pi[j] \models \varphi$
- $s \models \exists \varphi \mathsf{U} \psi$ iff for some $\pi \in APaths(s)$, there exists $0 \leq j < |\pi|$ such that $\pi[j] \models \psi$ and for all $0 \leq k < j$, $\pi[j] \models \varphi$

Proposition 2. *If $F = S$, for all $\varphi \in CTL$ and $s \in S$,*

- $[\![\varphi]\!](s) = \top$ *iff $s \models \varphi$, and*
- $[\![\varphi]\!](s) = \perp$ *iff $s \not\models \varphi$.*

Proof. Let $s \in S$. We prove this proposition by structural induction on φ. We distinguish the following cases.

- For the CTL formula a we have that

$$[\![a]\!](s) = \top \text{ iff } a \in L(s) \text{ iff } s \models a$$

and

$$[\![a]\!](s) = \perp \text{ iff } a \notin L(s) \text{ iff } s \not\models a.$$

- For the CTL formula $\neg\varphi$ we have that

$$\begin{aligned} & [\![\neg\varphi]\!](s) = \top \\ \text{iff } & [\![\varphi]\!](s) = \perp \\ \text{iff } & s \not\models \varphi \qquad \text{[by induction]} \\ \text{iff } & s \models \neg\varphi \end{aligned}$$

and

$$\begin{aligned} & [\![\neg\varphi]\!](s) = \perp \\ \text{iff } & [\![\varphi]\!](s) = \top \\ \text{iff } & s \models \varphi \qquad \text{[by induction]} \\ \text{iff } & s \not\models \neg\varphi \end{aligned}$$

- For the CTL formula $\varphi \wedge \psi$ we have that

$$\begin{aligned} & [\![\varphi \wedge \psi]\!](s) = \top \\ \text{iff } & [\![\varphi]\!](s) = \top \text{ and } [\![\psi]\!](s) = \top \\ \text{iff } & s \models \varphi \text{ and } s \models \psi \qquad \text{[by induction]} \\ \text{iff } & s \models \varphi \wedge \psi \end{aligned}$$

and

$$
\begin{aligned}
& [\![\varphi \wedge \psi]\!](s) = \perp \\
\text{iff } & [\![\varphi]\!](s) = \perp \text{ or } [\![\psi]\!](s) = \perp \\
\text{iff } & s \not\models \varphi \text{ or } s \not\models \psi && \text{[by induction]} \\
\text{iff } & s \not\models \varphi \wedge \psi
\end{aligned}
$$

- For the CTL formula $\forall \bigcirc \varphi$ we have that

$$
\begin{aligned}
& [\![\forall \bigcirc \varphi]\!](s) = \top \\
\text{iff } & \text{for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \top \\
\text{iff } & \text{for all } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \top && \text{[Proposition 1]} \\
\text{iff } & \text{for all } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, \pi[1] \models \varphi && \text{[by induction]} \\
\text{iff } & s \models \forall \bigcirc \varphi
\end{aligned}
$$

and

$$
\begin{aligned}
& [\![\forall \bigcirc \varphi]\!](s) = \perp \\
\text{iff } & \text{for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \perp \\
\text{iff } & \text{for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, \pi[1] \not\models \varphi && \text{[by induction]} \\
\text{iff } & s \not\models \forall \bigcirc \varphi
\end{aligned}
$$

- For the CTL formula $\exists \bigcirc \varphi$ we have that

$$
\begin{aligned}
& [\![\exists \bigcirc \varphi]\!](s) = \top \\
\text{iff } & \text{for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \top \\
\text{iff } & \text{for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, \pi[1] \models \varphi && \text{[by induction]} \\
\text{iff } & s \models \exists \bigcirc \varphi
\end{aligned}
$$

and

$$
\begin{aligned}
& [\![\exists \bigcirc \varphi]\!](s) = \perp \\
\text{iff } & \text{for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \perp \\
\text{iff } & \text{for all } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, [\![\varphi]\!](\pi[1]) = \perp && \text{[Proposition 1]} \\
\text{iff } & \text{for all } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1, \pi[1] \not\models \varphi && \text{[by induction]} \\
\text{iff } & s \not\models \exists \bigcirc \varphi
\end{aligned}
$$

- For the CTL formula $\forall \varphi \, \mathsf{U} \, \psi$ we have that details still need to be added here.
- For the CTL formula $\exists \varphi \, \mathsf{U} \, \psi$ we have that details still need to be added here.

$\square$

Next, we define equivalence of CTL formulas.

Definition 6. For CTL formulas φ and ψ,

$$\varphi \equiv \psi \text{ if } [\![\varphi]\!](s) = [\![\psi]\!](s) \text{ for all transition systems and } s \in S.$$

We use 2^S to denote the powerset of S. For each CTL formula we define the following two sets of states.

Definition 7. Let φ be a CTL formula. Then

$$\begin{aligned} Sat(\varphi) &= \{\, s \in S \mid [\![\varphi]\!](s) = \top \,\} \\ Unsat(\varphi) &= \{\, s \in S \mid [\![\varphi]\!](s) = \perp \,\} \end{aligned}$$

Theorem 1. *Let $(S, F, \rightarrow, I, AP, L)$ be a partial transition system. For all $a \in AP$ and φ, $\psi \in CTL$,*

- $$\begin{aligned} Sat(a) &= \{\, s \in S \mid a \in L(s) \,\} \\ Unsat(a) &= \{\, s \in S \mid a \notin L(s) \,\} \end{aligned}$$

- $$\begin{aligned} Sat(\texttt{true}) &= S \\ Unsat(\texttt{true}) &= \emptyset \end{aligned}$$

- $$\begin{aligned} Sat(\texttt{false}) &= \emptyset \\ Unsat(\texttt{false}) &= S \end{aligned}$$

- $$\begin{aligned} Sat(\neg\varphi) &= Unsat(\varphi) \\ Unsat(\neg\varphi) &= Sat(\varphi) \end{aligned}$$

- $$\begin{aligned} Sat(\varphi \wedge \psi) &= Sat(\varphi) \cap Sat(\psi) \\ Unsat(\varphi \wedge \psi) &= Unsat(\varphi) \cup Unsat(\psi) \end{aligned}$$

- $$\begin{aligned} Sat(\varphi \vee \psi) &= Sat(\varphi) \cup Sat(\psi) \\ Unsat(\varphi \vee \psi) &= Unsat(\varphi) \cap Unsat(\psi) \end{aligned}$$

- $$\begin{aligned} Sat(\varphi \to \psi) &= Unsat(\varphi) \cup Sat(\psi) \\ Unsat(\varphi \to \psi) &= Unsat(\varphi) \setminus Unsat(\psi) \end{aligned}$$

- $$\begin{aligned} Sat(\varphi \leftrightarrow \psi) &= \\ Unsat(\varphi \leftrightarrow \psi) &= \end{aligned}$$

- $$Sat(\forall \bigcirc \varphi) = \begin{cases} S & \textit{if } \varphi \equiv \texttt{true} \\ \{\, s \in F \mid post(s) \subseteq Sat(\varphi) \,\} & \textit{otherwise} \end{cases}$$
  $$Unsat(\forall \bigcirc \varphi) = \{\, s \in S \mid post(s) \cap Unsat(\varphi) \neq \emptyset \,\}$$

- $$Sat(\exists \bigcirc \varphi) = \begin{cases} S \setminus F \cup \{\, s \in F \mid post(s) \neq \emptyset \,\} & \textit{if } \varphi \equiv \texttt{true} \\ \{\, s \in S \mid post(s) \cap Sat(\varphi) \neq \emptyset \,\} & \textit{otherwise} \end{cases}$$
  $$Unsat(\exists \bigcirc \varphi) = \{\, s \in F \mid post(s) \subseteq Unsat(\varphi) \,\}$$

- $$\begin{aligned} Sat(\forall \varphi \mathsf{U} \psi) &= \\ Unsat(\forall \varphi \mathsf{U} \psi) &= \end{aligned}$$

- $$\begin{aligned} Sat(\exists \varphi \mathsf{U} \psi) &= \\ Unsat(\exists \varphi \mathsf{U} \psi) &= \end{aligned}$$

- *$Sat(\forall \Box \varphi)$ is the largest $U \subseteq S$ satisfying*
  $$U \subseteq \begin{cases} S & \textit{if } \varphi \equiv \texttt{true} \\ \{\, s \in F \mid s \in Sat(\varphi) \textit{ and } post(s) \subseteq U \,\} & \textit{otherwise} \end{cases}$$
  $Unsat(\forall \Box \varphi)$ is the smallest $V \subseteq S$ satisfying
  $$V \supseteq \{\, s \in S \mid s \in Unsat(\varphi) \textit{ or } post(s) \cap V \neq \emptyset \,\}$$

- $$Sat(\exists\Box\varphi) =$$
$$Unsat(\exists\Box\varphi) =$$

- $$Sat(\forall\Diamond\varphi) =$$
$$Unsat(\forall\Diamond\varphi) =$$

- $$Sat(\exists\Diamond\varphi) =$$
$$Unsat(\exists\Diamond\varphi) =$$

Proof. The first eight cases follow immediately from the definitions. For example, for the case $\varphi \wedge \psi$ we have that

$$\begin{aligned} Sat(\varphi \wedge \psi) &= \{\, s \in S \mid [\![\varphi \wedge \psi]\!](s) = \top \,\} \\ &= \{\, s \in S \mid [\![\varphi]\!](s) = \top \text{ and } [\![\psi]\!](s) = \top \,\} \\ &= \{\, s \in S \mid [\![\varphi]\!](s) = \top \,\} \cap \{\, s \in S \mid [\![\psi]\!](s) = \top \,\} \\ &= Sat(\varphi) \cap Sat(\psi) \end{aligned}$$

and

$$\begin{aligned} Unsat(\varphi \wedge \psi) &= \{\, s \in S \mid [\![\varphi \wedge \psi]\!](s) = \perp \,\} \\ &= \{\, s \in S \mid [\![\varphi]\!](s) = \perp \text{ or } [\![\psi]\!](s) = \perp \,\} \\ &= \{\, s \in S \mid [\![\varphi]\!](s) = \perp \,\} \cup \{\, s \in S \mid [\![\psi]\!](s) = \perp \,\} \\ &= Unsat(\varphi) \cup Unsat(\psi) \end{aligned}$$

- Consider the CTL formula $\forall \bigcirc \varphi$. For Sat, we distinguish two cases.
  - Assume that $\varphi \equiv \texttt{true}$. Then

$$\begin{aligned} Sat(\forall \bigcirc \varphi) &= \{\, s \in S \mid [\![\forall \bigcirc \varphi]\!](s) = \top \,\} \\ &= \{\, s \in S \mid [\![\varphi]\!](\pi[1]) = \top \text{ for all } \pi \in Paths(s) \text{ with } |\pi| > 1 \,\} \\ &= \{\, s \in S \mid [\![\texttt{true}]\!](\pi[1]) = \top \text{ for all } \pi \in Paths(s) \text{ with } |\pi| > 1 \,\} && [\varphi \equiv \texttt{true}] \\ &= S \end{aligned}$$

  - Otherwise, $\varphi \not\equiv \texttt{true}$. Then there exists $s_\perp \in \mathcal{S} \setminus S$ such that $[\![\varphi]\!](s_\perp) \neq \top$. We distinguish two cases.

- $*$ Assume that $s \in S \backslash F$. Then $ss_\bot \in \mathit{PPaths}(s)$. Since $[\![\varphi]\!]((ss_\bot)[1]) = [\![\varphi]\!](s_\bot) \neq \top$, we can conclude that $[\![\varphi]\!](\pi[1]) \neq \top$ for some $\pi \in \mathit{Paths}(s)$ with $|\pi| > 1$.
- $*$ Assume $s \in F$. Then $\mathit{post}(s) = \{\, \pi[1] \mid \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \,\}$.

From the above, we can conclude that

$$\begin{aligned}
\mathit{Sat}(\forall \bigcirc \varphi) =& \{\, s \in S \mid [\![\forall \bigcirc \varphi]\!](s) = \top \,\} \\
=& \{\, s \in S \mid [\![\varphi]\!](\pi[1]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \,\} \\
=& \{\, s \in S \setminus F \mid [\![\varphi]\!](\pi[1]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \,\} \cup \\
& \{\, s \in F \mid [\![\varphi]\!](\pi[1]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ with } |\pi| > 1 \,\} \\
=& \emptyset \cup \{\, s \in F \mid [\![\varphi]\!](s') = \top \text{ for all } s' \in \mathit{post}(s) \,\} \\
=& \{\, s \in F \mid \mathit{post}(s) \subseteq \mathit{Sat}(\varphi) \,\}
\end{aligned}$$

Next, we consider Unsat.

$$\begin{aligned}
\mathit{Unsat}(\forall \bigcirc \varphi) =& \{\, s \in S \mid [\![\forall \bigcirc \varphi]\!](s) = \bot \,\} \\
=& \{\, s \in S \mid [\![\varphi]\!](\pi[1]) = \bot \text{ for some } \pi \in \mathit{APaths}(s) \text{ with } |\pi| > 1 \,\} \\
=& \{\, s \in S \mid [\![\varphi]\!](s') = \bot \text{ for some } s' \in \mathit{post}(s) \,\} \\
=& \{\, s \in S \mid \mathit{post}(s) \cap \mathit{Unsat}(\varphi) \neq \emptyset \,\}
\end{aligned}$$

- Who wants to try $\exists \bigcirc \varphi$?
- Consider the CTL formula $\forall \Box \varphi$. We first focus on Sat.

  Given the CTL formula φ, the function

$$\mathcal{F}_\varphi : 2^S \to 2^S$$

  is defined by

$$\mathcal{F}_\varphi(U) = \begin{cases} S & \text{if } \varphi \equiv \texttt{true} \\ \{\, s \in F \mid s \in \mathit{Sat}(\varphi) \text{ and } \mathit{post}(s) \subseteq U \,\} & \text{otherwise} \end{cases}$$

  Next, we show that the function $\mathcal{F}_\varphi$ is monotone, that is, for all $U, V \in 2^S$, if $U \subseteq V$ then $\mathcal{F}_\varphi(U) \subseteq \mathcal{F}_\varphi(V)$. Let $U, V \in 2^S$ and assume that $U \subseteq V$. Obviously, $\mathcal{F}_\varphi(U) \subseteq \mathcal{F}_\varphi(V)$ if $\varphi \equiv \texttt{true}$. Otherwise, let $s \in \mathcal{F}_\varphi(U)$. To conclude that $\mathcal{F}_\varphi(U) \subseteq \mathcal{F}_\varphi(V)$, it remains to show that $s \in \mathcal{F}_\varphi(V)$. Since $s \in \mathcal{F}_\varphi(U)$, we have that $s \in F$, $s \in \mathit{Sat}(\varphi)$, and $\mathit{post}(s) \subseteq U$. Since $U \subseteq V$, we can conclude that $\mathit{post}(s) \subseteq V$. Hence, $s \in \mathcal{F}_\varphi(V)$. From the Knaster-Tarski theorem we can conclude that there exists a largest $U \subseteq S$ satisfying $U \subseteq \mathcal{F}_\varphi(U)$.

  We distinguish two cases.

– Assume that $\varphi \equiv \texttt{true}$. Then

$$\begin{aligned}
\mathit{Sat}(\forall\Box\varphi) &= \{\, s \in S \mid [\![\forall\Box\varphi]\!](s) = \top \,\} \\
&= \{\, s \in S \mid [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \,\} \\
&= \{\, s \in S \mid [\![\texttt{true}]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \,\} \qquad [\varphi \equiv \texttt{true}] \\
&= S
\end{aligned}$$

– Otherwise, $\varphi \not\equiv \texttt{true}$. Then there exists $s_\bot \in \mathcal{S} \setminus S$ such that $[\![\varphi]\!](s_\bot) \neq \top$. We distinguish two cases.

  * Assume that $s \in S \backslash F$. Then $ss_\bot \in \mathit{PPaths}(s)$. Since $[\![\varphi]\!]((ss_\bot)[1]) = [\![\varphi]\!](s_\bot) \neq \top$, we can conclude that $[\![\varphi]\!](\pi[1]) \neq \top$ for some $\pi \in \mathit{Paths}(s)$ with $|\pi| > 1$.
  * Assume $s \in F$. Then $\mathit{Paths}(s) = \{\, s\pi \mid s' \in \mathit{post}(s) \text{ and } \pi \in \mathit{Paths}(s') \,\}$.

  From the above we can conclude that

$$\begin{aligned}
\mathit{Sat}(\forall\Box\varphi) =&\{\, s \in S \mid [\![\forall\Box\varphi]\!](s) = \top \,\} \\
=&\{\, s \in S \mid [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \,\} \\
=&\{\, s \in S \setminus F \mid [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \,\} \cup \\
=&\{\, s \in F \mid [\![\varphi]\!](\pi[i]) = \top \text{ for all } \pi \in \mathit{Paths}(s) \text{ and } 0 \leq i < |\pi| \,\} \\
=&\emptyset \cup \{\, s \in F \mid [\![\varphi]\!]((s\pi)[i]) = \top \text{ for all } s' \in \mathit{post}(s) \text{ and } \pi \in \mathit{Paths}(s') \text{ and } 0 \leq i < |\pi| \,\} \\
=&\{\, s \in F \mid [\![\varphi]\!](s) = \top \text{ and } [\![\varphi]\!](\pi[i]) = \top \text{ for all } s' \in \mathit{post}(s) \text{ and } \pi \in \mathit{Paths}(s') \text{ and } 0 \leq \\
=&\{\, s \in F \mid s \in \mathit{Sat}(\varphi) \text{ and } s' \in \mathit{Sat}(\forall\Box\varphi) \text{ for all } s' \in \mathit{post}(s) \,\} \\
=&\{\, s \in F \mid s \in \mathit{Sat}(\varphi) \text{ and } \mathit{post}(s) \subseteq \mathit{Sat}(\forall\Box\varphi) \,\}
\end{aligned}$$

Let $U \subseteq S$ satisfying $U \subseteq \mathcal{F}_\varphi(U)$. It remains to show that $U \subseteq \mathit{Sat}(\forall\Box\varphi)$. We distinguish two cases.

– If $\varphi \equiv \texttt{true}$ then $U \subseteq \mathcal{F}_\varphi(U) = S = \mathit{Sat}(\forall\Box\varphi)$.

– Otherwise, $\varphi \not\equiv \texttt{true}$. Towards a contradiction, assume that $s \in U$ and $s \notin \mathit{Sat}(\forall\Box\varphi)$. Then $[\![\forall\Box\varphi]\!](s) \neq \top$. Hence, there exists $\pi \in \mathit{Paths}(s)$ and $0 \leq i < |\pi|$ such that $[\![\varphi]\!](\pi[i]) \neq \top$. Next, we show that $\pi[j] \in U$ for all $0 \leq j \leq i$ by induction on j. In the base case $j = 0$, we have that $\pi[0] = s \in U$. Assume that $j > 0$. By induction, $\pi[j-1] \in U$. Since $U \subseteq \mathcal{F}_\varphi(U)$, we have that $\pi[j-1] \in F$ and $\mathit{post}(\pi[j-1]) \subseteq U$. Hence, $\pi[j] \in U$. Since $\pi[i] \in U$ and $U \subseteq \mathcal{F}_\varphi(U)$, we can conclude that $\pi[i] \in \mathit{Sat}(\varphi)$, which contradicts $[\![\varphi]\!](\pi[i]) \neq \top$.

Next, we focus on Unsat. Given the CTL formula φ, the function

$$\mathcal{G}_\varphi : 2^S \to 2^S$$

is defined by

$$\mathcal{G}_\varphi(V) = \{\, s \in S \mid s \in \mathit{Unsat}(\varphi) \text{ or } \mathit{post}(s) \cap V \neq \emptyset \,\}$$

Next, we show that the function $\mathcal{G}_\varphi$ is monotone, that is, for all $U, V \in 2^S$, if $U \subseteq V$ then $\mathcal{G}_\varphi(U) \subseteq \mathcal{G}_\varphi(V)$. Let $U, V \in 2^S$ and assume that $U \subseteq V$. Let $s \in \mathcal{G}_\varphi(U)$. To conclude that $\mathcal{G}_\varphi(U) \subseteq \mathcal{G}_\varphi(V)$, it remains to show that $s \in \mathcal{G}_\varphi(V)$. Since $s \in \mathcal{G}_\varphi(U)$, we have that $s \in S$, $s \in \mathit{Unsat}(\varphi)$ or $\mathit{post}(s) \cap U \neq \emptyset$. Since $U \subseteq V$, we can conclude that $\mathit{post}(s) \cap V \neq \emptyset$. Hence, $s \in \mathcal{G}_\varphi(V)$. From the Knaster-Tarski theorem we can conclude that there exists a smallest $V \subseteq S$ satisfying $V \supseteq \mathcal{F}_\varphi(V)$.

Since $\mathit{APaths}(s) = \{s\} \cup \{\, s\pi \mid s' \in \mathit{post}(s) \text{ and } \pi \in \mathit{APaths}(s') \,\}$, we have that

$$\begin{aligned}
\mathit{Unsat}(\forall\Box\varphi) &= \{\, s \in S \mid [\![\forall\Box\varphi]\!](s) = \bot \,\} \\
&= \{\, s \in S \mid [\![\varphi]\!](\pi[i]) = \bot \text{ for some } \pi \in \mathit{APaths}(s) \text{ and } 0 \leq i < |\pi| \,\} \\
&= \{\, s \in S \mid [\![\varphi]\!](s) = \bot \text{ or } [\![\varphi]\!](\pi[i]) = \bot \text{ for some } s' \in \mathit{post}(s) \text{ and } \pi \in \mathit{APaths}(s') \text{ and } 0 \leq \\
&= \{\, s \in S \mid s \in \mathit{Unsat}(\varphi) \text{ or } \mathit{post}(s) \cap \mathit{Unsat}(\forall\Box\varphi) \neq \emptyset \,\}
\end{aligned}$$

Let $V \subseteq S$ satisfying $V \supseteq \mathcal{G}_\varphi(V)$. It remains to show that $V \supseteq \mathit{Unsat}(\forall\Box\varphi)$. Towards a contradiction, assume that $s \in \mathit{Unsat}(\forall\Box\varphi)$ and $s \notin V$. Then $[\![\varphi]\!](\pi[i]) = \bot$ for some $\pi \in \mathit{APaths}(s)$ and $0 \leq i < |\pi|$. A few more details need to be added (soon).

- Who wants to try $\exists\Box\varphi$?

$\Box$

We first consider the formula $\forall\bigcirc\varphi$, where φ is some arbitrary formula. Assume that the state s is fully explored. In that case,

$$[\![\forall\bigcirc\varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s') = \top \text{ for all } s \to s' \\ \bot & \text{if } [\![\varphi]\!](s') = \bot \text{ for some } s \to s' \\ ? & \text{otherwise.} \end{cases}$$

The above is equivalent to

$$[\![\forall\bigcirc\varphi]\!](s) = \bigwedge\{\, [\![\varphi]\!](s') \mid s \to s' \,\}.$$

Note that $\bigwedge\emptyset = \top$.

Next, assume that the state s is not fully explored. Recall that state s has outgoing transitions, but those have not been explored yet. We distinguish three cases.

- None of outgoing transitions of state s have been explored yet. Let us first consider the case that φ is `true`. In that case, $[\![\forall\bigcirc\,\texttt{true}]\!](s) = \top$ because every state satisfies `true`. Similarly, we have that $[\![\forall\bigcirc\,(\texttt{true} \vee \texttt{false})]\!](s) = \top$. To capture all these cases, we use the notion of equivalence of CTL formulas, denoted $\equiv$ (see [1, Definition 6.12]). If $\varphi \equiv \texttt{true}$ then $[\![\forall\bigcirc\varphi]\!](s) = \top$.

  Next, let us consider the case that φ is `false`. No state satisfies `false`. Hence, $[\![\forall\bigcirc\,\texttt{false}]\!](s) = \bot$. More generally, $\varphi \equiv \texttt{false}$ then $[\![\forall\bigcirc\varphi]\!](s) = \bot$.

Otherwise, $\varphi \not\equiv \texttt{true}$ and $\varphi \not\equiv \texttt{false}$. The former implies that there exists a transition system with initial state $s_{\texttt{true}}$ such that $s_{\texttt{true}} \not\models \varphi$ and the latter implies that there exists a transition system with initial state $s_{\texttt{false}}$ such that $s_{\texttt{false}} \models \varphi$. State s_0 has at least one outgoing transition. We do not know anything about the target state of that transition (it could be $s_{\texttt{true}}$ or $s_{\texttt{false}}$ or any other state), we do not know that value of $[\![\forall \bigcirc \varphi]\!](s)$.

Combining the above, we arrive at

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

- State s has a single explored outgoing transition to state s_1. State s has more outgoing transitions, but those have not been explored yet. As in the previous case, if $\varphi \equiv \texttt{true}$ then $[\![\forall \bigcirc \varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\forall \bigcirc \varphi]\!](s) = \bot$.

  Now let us consider state s_1 and formula φ. If $[\![\varphi]\!](s_1) = \bot$, we can conclude that $[\![\forall \bigcirc \varphi]\!](s) = \bot$. In case $[\![\varphi]\!](s) \neq \bot$, we obtain no additional information about $[\![\forall \bigcirc \varphi]\!](s)$.

  Combining the above, we arrive at

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s_1) = \bot \\ ? & \text{otherwise.} \end{cases}$$

  Note that $[\![\varphi]\!](s_1) = \bot$ if $\varphi \equiv \texttt{false}$. Hence, we can simplify the above to

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s_1) = \bot \\ ? & \text{otherwise.} \end{cases}$$

- State s has explored outgoing transitions to state s_1 and s_2. State s has more outgoing transitions, but those have not been explored yet. As in the previous cases, if $\varphi \equiv \texttt{true}$ then $[\![\forall \bigcirc \varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\forall \bigcirc \varphi]\!](s) = \bot$.

  Now let us consider state s_1 and s_2. If $[\![\varphi]\!](s_1) = \bot$ or $[\![\varphi]\!](s_2) = \bot$, we can conclude that $[\![\forall \bigcirc \varphi]\!](s) = \bot$. In case $[\![\varphi]\!](s_1) \neq \bot$ and $[\![\varphi]\!](s_2) \neq \bot$, we obtain no additional information about $[\![\forall \bigcirc \varphi]\!](s)$. Therefore,

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s_1) = \bot \text{ or } [\![\varphi]\!](s_2) = \bot \\ ? & \text{otherwise.} \end{cases}$$

  This can be simplified to

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s_1) = \bot \text{ or } [\![\varphi]\!](s_2) = \bot \\ ? & \text{otherwise.} \end{cases}$$

The above three cases can be generalized as follows.

$$[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s') = \bot \text{ for some } s \rightarrow s' \\ ? & \text{otherwise.} \end{cases}$$

This can be reformulated as follows.

$$\begin{aligned} \mathit{Sat}(\forall \bigcirc \varphi) &= \begin{cases} S & \text{if } \varphi \equiv \texttt{true} \\ \{ s \in F \mid \mathit{post}(s) \subseteq \mathit{Sat}(\varphi) \} & \text{otherwise} \end{cases} \\ \mathit{Unsat}(\forall \bigcirc \varphi) &= \begin{cases} (S \setminus F) \cup \{ s \in F \mid \mathit{post}(s) \neq \emptyset \} & \text{if } \varphi \equiv \texttt{false} \\ \{ s \in S \mid \mathit{post}(s) \cap \mathit{Unsat}(\varphi) \neq \emptyset \} & \text{otherwise} \end{cases} \\ \mathit{Unknown}(\forall \bigcirc \varphi) &= S \setminus (\mathit{Sat}(\forall \bigcirc \varphi) \cup \mathit{Unsat}(\forall \bigcirc \varphi)) \end{aligned}$$

In the above characterization we use

$$\mathit{post}(s) = \{ s' \in S \mid s \rightarrow s' \}.$$

Next, let us consider the formula $\exists \bigcirc \varphi$, where φ is some arbitary formula. Assume that the state s is fully explored. In that case,

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s') = \top \text{ for some } s \rightarrow s' \\ \bot & \text{if } [\![\varphi]\!](s') = \bot \text{ for all } s \rightarrow s' \\ ? & \text{otherwise.} \end{cases}$$

The above is equivalent to

$$[\![\exists \bigcirc \varphi]\!](s) = \bigvee \{ [\![\varphi]\!](s') \mid s \rightarrow s' \}.$$

Note that $\bigvee \emptyset = \bot$.

Next, assume that the state s is not fully explored. Recall that state s has outgoing transitions, but those have not been explored yet. We distinguish three cases.

- None of outgoing transitions of state s have been explored yet. Let us first consider the case that φ is `true`. In that case, $[\![\exists \bigcirc \texttt{true}]\!](s) = \top$ because state s has outgoing transitions and every state satisfies `true`. More generally, if $\varphi \equiv \texttt{true}$ then $[\![\exists \bigcirc \varphi]\!](s) = \top$.

  Next, let us consider the case that φ is `false`. No state satisfies `false`. Hence, $[\![\exists \bigcirc \texttt{false}]\!](s) = \bot$. More generally, $\varphi \equiv \texttt{false}$ then $[\![\exists \bigcirc \varphi]\!](s) = \bot$.

  Otherwise, $\varphi \not\equiv \texttt{true}$ and $\varphi \not\equiv \texttt{false}$. The former implies that there exists a transition system with initial state $s_{\texttt{true}}$ such that $s_{\texttt{true}} \not\models \varphi$ and the latter implies that there exists a transition system with initial state $s_{\texttt{false}}$ such that $s_{\texttt{false}} \models \varphi$. State s has at least one outgoing transition. We do not know anything about the target state of that transition (it could be $s_{\texttt{true}}$ or $s_{\texttt{false}}$ or any other state), we do not know that value of $[\![\exists \bigcirc \varphi]\!](s)$.

  Combining the above, we arrive at

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

- State s has a single explored outgoing transition to state s_1. State s has more outgoing transitions, but those have not been explored yet. As in the previous case, if $\varphi \equiv \texttt{true}$ then $[\![\exists \bigcirc \varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\exists \bigcirc \varphi]\!](s) = \perp$.

  Now let us consider state s_1 and formula φ. If $[\![\varphi]\!](s_1) = \top$, we can conclude that $[\![\exists \bigcirc \varphi]\!](s) = \top$. In case $[\![\varphi]\!](s_1) \neq \top$, we obtain no additional information about $[\![\exists \bigcirc \varphi]\!](s)$.

  Combining the above, we arrive at

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \text{ or } [\![\varphi]\!](s_1) = \top \\ \perp & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

  Note that $[\![\varphi]\!](s_1) = \top$ if $\varphi \equiv \texttt{true}$. Hence, we can simplify the above to

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s_1) = \top \\ \perp & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

- State s has explored outgoing transitions to state s_1 and s_2. State s has more outgoing transitions, but those have not been explored yet. As in the previous cases, if $\varphi \equiv \texttt{true}$ then $[\![\exists \bigcirc \varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\exists \bigcirc \varphi]\!](s) = \perp$.

  Now let us consider state s_1 and s_2. If $[\![\varphi]\!](s_1) = \top$ or $[\![\varphi]\!](s_2) = \top$, we can conclude that $[\![\exists \bigcirc \varphi]\!](s) = \top$. In case $[\![\varphi]\!](s_1) \neq \top$ and $[\![\varphi]\!](s_2) \neq \top$, we obtain no additional information about $[\![\exists \bigcirc \varphi]\!](s)$. Therefore,

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \text{ or } [\![\varphi]\!](s_1) = \top \text{ or } [\![\varphi]\!](s_2) = \top \\ \perp & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

  This can be simplified to

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s_1) = \top \text{ or } [\![\varphi]\!](s_2) = \top \\ \perp & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

The above three cases can be generalized as follows.

$$[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \text{ or } [\![\varphi]\!](s') = \top \text{ for some } s \to s' \\ \perp & \text{if } \varphi \equiv \texttt{false} \\ ? & \text{otherwise.} \end{cases}$$

This can be reformulated as follows.

$$\begin{aligned} Sat(\exists \bigcirc \varphi) &= \begin{cases} S \setminus F \cup \{\, s \in F \mid post(s) \neq \emptyset \,\} & \text{if } \varphi \equiv \texttt{true} \\ \{\, s \in S \mid post(s) \cap Sat(\varphi) \neq \emptyset \,\} & \text{otherwise} \end{cases} \\ Unsat(\exists \bigcirc \varphi) &= \begin{cases} S & \text{if } \varphi \equiv \texttt{false} \\ \{\, s \in F \mid post(s) \subseteq Unsat(\varphi) \,\} & \text{otherwise} \end{cases} \\ Unknown(\exists \bigcirc \varphi) &= S \setminus (Sat(\exists \bigcirc \varphi) \cup Unsat(\exists \bigcirc \varphi)) \end{aligned}$$

Next, let us consider the formulas $\forall\Box\varphi$. Assume that the state s is fully explored. Then

$$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s) = \top \text{ and } [\![\forall\Box\varphi]\!](s') = \top \text{ for all } s \to s' \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s') = \bot \text{ for some } s \to s' \\ ? & \text{otherwise.} \end{cases}$$

Next, assume that the state s is not fully explored. Recall that state s has outgoing transitions, but those have not been explored yet. We distinguish three cases.

- None of outgoing transitions of state s have been explored yet. Let us first consider the case that φ is `true`. In that case, $[\![\forall\Box\texttt{true}]\!](s) = \top$ because each (reachable) state satisfies `true`.

  Next, let us consider the case that φ is `false`. No state satisfies `false`. Hence, $[\![\forall\Box\texttt{false}]\!](s) = \bot$. More generally, $\varphi \equiv \texttt{false}$ then $[\![\forall\Box\varphi]\!](s) = \bot$.

  If $[\![\varphi]\!](s) = \bot$, there exists a state reachable from state s that does not satisfy φ and, hence, $[\![\forall\Box\varphi]\!](s) = \bot$.

  Otherwise, $\varphi \not\equiv \texttt{true}$ and $\varphi \not\equiv \texttt{false}$. The former implies that there exists a transition system with initial state $s_{\texttt{true}}$ such that $s_{\texttt{true}} \not\models \varphi$ and the latter implies that there exists a transition system with initial state $s_{\texttt{false}}$ such that $s_{\texttt{false}} \models \varphi$. Hence, $[\![\forall\Box\varphi]\!](s) = ?$.

  Combining the above, we arrive at

  $$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s) = \bot \\ ? & \text{otherwise.} \end{cases}$$

  This can be further simplified to

  $$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \\ ? & \text{otherwise.} \end{cases}$$

- State s has a single explored outgoing transition to state s_1. State s has more outgoing transitions, but those have not been explored yet. As in the previous case, if $\varphi \equiv \texttt{true}$ then $[\![\forall\Box\varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\forall\Box\varphi]\!](s) = \bot$.

  If $[\![\varphi]\!](s) = \bot$, there exists a state reachable from state s that does not satisfy φ and, hence, $[\![\forall\Box\varphi]\!](s) = \bot$.

  Now let us consider state s_1. If $[\![\forall\Box\varphi]\!](s_1) = \bot$, we can conclude that $[\![\forall\Box\varphi]\!](s) = \bot$. In case $[\![\forall\Box\varphi]\!](s_1) \neq \bot$, we obtain no additional information about $[\![\forall\Box\varphi]\!](s)$.

  Combining the above, we arrive at

  $$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_1) = \bot \\ ? & \text{otherwise.} \end{cases}$$

This can be further simplified to

$$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_1) = \bot \\ ? & \text{otherwise.} \end{cases}$$

- State s has explored outgoing transitions to state s_1 and s_2. State s has more outgoing transitions, but those have not been explored yet. As in the previous cases, if $\varphi \equiv \texttt{true}$ then $[\![\forall\Box\varphi]\!](s) = \top$, and if $\varphi \equiv \texttt{false}$ then $[\![\forall\Box\varphi]\!](s) = \bot$.

  If $[\![\varphi]\!](s) = \bot$, there exists a state reachable from state s that does not satisfy φ and, hence, $[\![\forall\Box\varphi]\!](s) = \bot$.

  Now let us consider state s_1 and s_2. If $[\![\forall\Box\varphi]\!](s_1) = \bot$, we can conclude that $[\![\forall\Box\varphi]\!](s) = \bot$. Similarly, if $[\![\forall\Box\varphi]\!](s_2) = \bot$, then $[\![\forall\Box\varphi]\!](s) = \bot$. Otherwise, we obtain no additional information about $[\![\forall\Box\varphi]\!](s)$. Therefore,

$$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } \varphi \equiv \texttt{false} \text{ or } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_1) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_2) = \bot \\ ? & \text{otherwise.} \end{cases}$$

  This can be further simplified to

$$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_1) = \bot \text{ or } [\![\forall\Box\varphi]\!](s_2) = \bot \\ ? & \text{otherwise.} \end{cases}$$

The above three cases can be generalized as follows.

$$[\![\forall\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \texttt{true} \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \text{ or } [\![\forall\Box\varphi]\!](s') = \bot \text{ for some } s \to s' \\ ? & \text{otherwise.} \end{cases}$$

Now consider that case that state s is not fully explored and transitions to itself and assume that $\varphi \not\equiv \texttt{false}$ and $[\![\varphi]\!](s) \neq \bot$. Then $[\![\forall\Box\varphi]\!](s) = \bot$ if $[\![\forall\Box\varphi]\!](s) = \bot$. To demonstrate that $[\![\forall\Box\varphi]\!]$ can be defined so that it satisfies the above equation, we will rely on the Knaster-Tarski fixed-point theorem [5, 8].

Let us first reformulate the above in terms of Sat, Unsat, and $\mathit{Unknown}$.

$$\mathit{Sat}(\forall\Box\varphi) = \begin{cases} S & \text{if } \varphi \equiv \texttt{true} \\ \{\, s \in F \mid s \in \mathit{Sat}(\varphi) \wedge \mathit{post}(s) \subseteq \mathit{Sat}(\forall\Box\varphi) \,\} & \text{otherwise} \end{cases}$$

$$\mathit{Unsat}(\forall\Box\varphi) = \begin{cases} (S \setminus F) \cup \{\, s \in F \mid \mathit{post}(s) \neq \emptyset \,\} & \text{if } \varphi \equiv \texttt{false} \\ \{\, s \in S \mid s \in \mathit{Unsat}(\varphi) \vee \mathit{post}(s) \cap \mathit{Unsat}(\forall\Box\varphi) \neq \emptyset \,\} & \text{otherwise} \end{cases}$$

$$\mathit{Unknown}(\forall\Box\varphi) = S \setminus (\mathit{Sat}(\forall\Box\varphi) \cup \mathit{Unsat}(\forall\Box\varphi))$$

To address the above observed circularity, we start with ordering sets of states, that is, the elements of 2^S. Clearly, we have that for all $U, V, W \in 2^S$,

- $U \subseteq U$,
- if $U \subseteq V$ and $V \subseteq U$ then $U = V$, and
- if $U \subseteq V$ and $V \subseteq W$ then $U \subseteq W$.

The set 2^S and the order $\subseteq$ form a complete lattice (see, for example, [4, Section 6.1]). Given a CTL formula φ, with $\varphi \not\equiv \texttt{true}$, the function

$$\mathcal{F}_\varphi : 2^S \to 2^S$$

is defined by

$$\mathcal{F}_\varphi(U) = \{\, s \in F \mid s \in \mathit{Sat}(\varphi) \wedge \mathit{post}(s) \subseteq U \,\}.$$

Next, we show that the function $\mathcal{F}_\varphi$ is monotone, that is, for all U, $V \in 2^S$, if $U \subseteq V$ then $\mathcal{F}_\varphi(U) \subseteq \mathcal{F}_\varphi(V)$. Let U, $V \in 2^S$ and assume that $U \subseteq V$. Let $s \in \mathcal{F}_\varphi(U)$. To conclude that $\mathcal{F}_\varphi(U) \subseteq \mathcal{F}_\varphi(V)$, it remains to show that $s \in \mathcal{F}_\varphi(V)$. Since $s \in \mathcal{F}_\varphi(U)$, we have that $s \in F$, $s \in \mathit{Sat}(\varphi)$, and $\mathit{post}(s) \subseteq U$. Since $U \subseteq V$, we can conclude that $\mathit{post}(s) \subseteq V$. Hence, $s \in \mathcal{F}_\varphi(V)$.

According to the Knaster-Tarski theorem, a monotone function on a complete lattice has a greatest fixed-point. As we already observed above, the set 2^S with the order $\subseteq$ is a complete lattice. Furthermore, we have shown that the function $\mathcal{F}_\varphi$ is monotone. Hence, $\mathcal{F}_\varphi$ has a greatest fixed point. A set $U \in 2^S$ is a fixed-point of $\mathcal{F}_\varphi$ if $\mathcal{F}_\varphi(U) = U$. A set $U \in 2^S$ is a greatest fixed-point of $\mathcal{F}_\varphi$ if U is a fixed-point of $\mathcal{F}_\varphi$ and for every fixed-point V of $\mathcal{F}_\varphi$ we have that $V \subseteq U$. We define $\mathit{Sat}(\forall\Box\varphi)$ as the greatest fixed-point of $\mathcal{F}_\varphi$.

Since the set 2^S is finite, the Knaster-Tarski fixed point theorem also suggests an algorithm to compute the greatest fixed-point of $\mathcal{F}_\varphi$ since this greatest fixed point is equal to

$$\bigcup \{\, U \in 2^S \mid U \subseteq \mathcal{F}_\varphi(U) \,\}.$$

According to, for example, [4, Section 6.2], we can compute $\mathit{Sat}(\forall\Box\varphi)$ as follows.

```
U = S
repeat
  U_old = U
  U = { s ∈ F | s ∈ Sat(φ) ∧ post(s) ⊆ U }
until U_old = U
```

Given a CTL formula φ, with $\varphi \not\equiv \texttt{false}$, the function

$$\mathcal{G}_\varphi : 2^S \to 2^S$$

is defined by

$$\mathcal{G}_\varphi(U) = \{\, s \in S \mid s \in \mathit{Unsat}(\varphi) \wedge \mathit{post}(s) \cap U \neq \emptyset \,\}.$$

Next, we show that the function $\mathcal{G}_\varphi$ is monotone, that is, for all U, $V \in 2^S$, if $U \subseteq V$ then $\mathcal{G}_\varphi(U) \subseteq \mathcal{G}_\varphi(V)$. Let U, $V \in 2^S$ and assume that $U \subseteq V$. Let $s \in \mathcal{G}_\varphi(U)$. To conclude that

$\mathcal{G}_\varphi(U) \subseteq \mathcal{G}_\varphi(V)$, it remains to show that $s \in \mathcal{G}_\varphi(V)$. Since $s \in \mathcal{G}_\varphi(U)$, we have that $s \in S$, $s \in \mathit{Unsat}(\varphi)$ and $\mathit{post}(s) \cap U \neq \emptyset$. Since $U \subseteq V$, we can conclude that $\mathit{post}(s) \cap V \neq \emptyset$. Hence, $s \in \mathcal{G}_\varphi(V)$.

According to the Knaster-Tarski theorem, a monotone function on a complete lattice has a least fixed-point. Since the set 2^S is finite, the Knaster-Tarski fixed point theorem also suggests an algorithm to compute the least fixed-point of $\mathcal{G}_\varphi$ since this least fixed point is equal to

$$\bigcap \{\, U \in 2^S \mid \mathcal{G}_\varphi(U) \subseteq U \,\}.$$

According to, for example, [4, Section 6.2], we can compute $\mathit{Unsat}(\forall\Box\varphi)$ as follows.

$U = \emptyset$
`repeat`
  $U_{old} = U$
  $U = \{\, s \in S \mid s \in \mathit{Unsat}(\varphi) \wedge \mathit{post}(s) \cap U \neq \emptyset \,\}$
`until` $U_{old} = U$

Next, let us consider the formulas $\exists\Box\varphi$. Assume that the state s is fully explored. Then

$$[\![\exists\Box\varphi]\!](s) = \begin{cases} \top & \text{if } [\![\varphi]\!](s) = \top \text{ and } (s \not\to \text{ or } [\![\exists\Box\varphi]\!](s') = \top \text{ for some } s \to s') \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \text{ or } (s \to \text{ and } [\![\exists\Box\varphi]\!](s') = \bot \text{ for all } s \to s') \\ ? & \text{otherwise.} \end{cases}$$

Next, assume that the state s is not fully explored. Then

$$[\![\exists\Box\varphi]\!](s) = \begin{cases} \top & \text{if } \varphi \equiv \mathtt{true} \text{ or } ([\![\varphi]\!](s) = \top \text{ and } [\![\exists\Box\varphi]\!](s') = \top \text{ for some } s \to s') \\ \bot & \text{if } [\![\varphi]\!](s) = \bot \\ ? & \text{otherwise.} \end{cases}$$

Let us first reformulate the above in terms of Sat, Unsat, and $\mathit{Unknown}$.

$$\begin{aligned} \mathit{Sat}(\exists\Box\varphi) =& \{\, s \in S \mid s \in \mathit{Sat}(\varphi) \wedge \mathit{post}(s) \cap \mathit{Sat}(\exists\Box\varphi) \neq \emptyset \,\} \cup \\ & \{\, s \in F \mid s \in \mathit{Sat}(\varphi) \wedge \mathit{post}(s) = \emptyset \,\} \\ \mathit{Unsat}(\exists\Box\varphi) =& \mathit{Unsat}(\varphi) \cup \{\, s \in F \mid \emptyset \neq \mathit{post}(s) \subseteq \mathit{Unsat}(\varphi) \,\} \\ \mathit{Unknown}(\exists\Box\varphi) =& S \setminus (\mathit{Sat}(\exists\Box\varphi) \cup \mathit{Unsat}(\exists\Box\varphi)) \end{aligned}$$

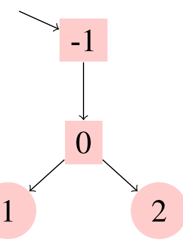
The set $\mathit{Sat}(\exists\Box\varphi)$ can be defined as a greatest fixed-point.

Similarly, we might be able to characterize $\mathit{Unsat}(\forall\Box\varphi)$ as a least fixed-point.

7 JPF Listener that Writes a Partial Transition System to File

A listener for Java PathFinder (JPF) writes its partial transition system to file. In [2, Section 7.3], a listener that writes a transition system to file has been developed. This listener is extended to the setting of partial transition systems.

Consider the following partial transition system.



State -1 is the initial state. The states -1 and 0 are fully explored, and states 1 and 2 are not fully explored. The listener produces a file, the name of which is the name of the system under test with ".tra" as suffix (see [2, Section 7.4]), with the following content.

```
-1 -> 0
0 -> 1
0 -> 2
1 2
```

The first three lines describe the transitions. Each line contains the source of the transition followed by the target of the transition. The last line contains the states that are not fully explored yet.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, USA, 2008.

[2] Franck van Breugel. Java PathFinder: a tool to detect bugs in Java code. 2020.

[3] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Proceedings of the 3rd Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, Yorktown Heights, NY, USA, May 1981. Springer-Verlag.

[4] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[5] Bronisław Knaster. Un théorème sur les fonctions d'ensembles. *Annales de la Société Polonaise de Mathématique*, 6:133–134, 1928.

[6] Terence Parr. *The Definitive ANTLR 4 Reference*. The Pragmatic Bookshelf, Dallas, TX, USA, 2013.

[7] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, RI, USA, October/November 1977. IEEE.

[8] Alfred Tarski. A lattice-theoretic fixed point theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, June 1955.

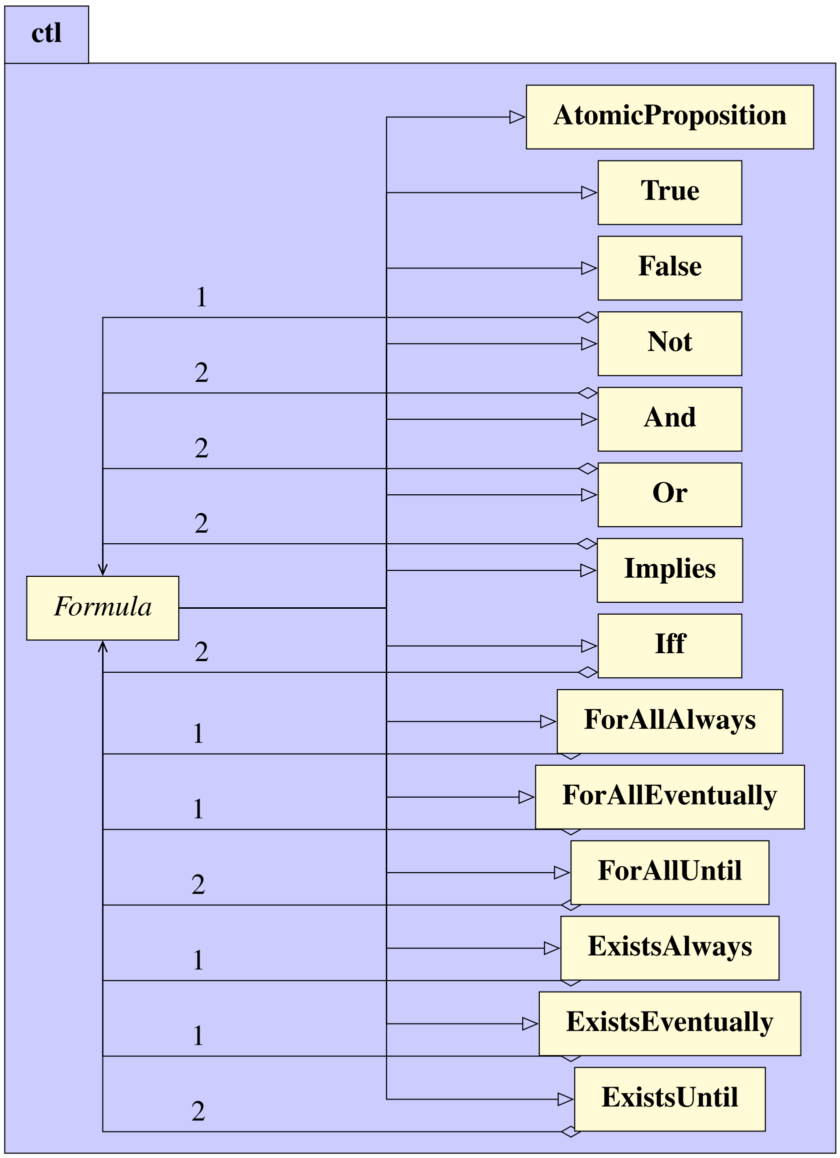


Figure 1: UML class diagram of the abstract syntax classes.