July 15, 2021

1 The Syntax of Computation Tree Logic

Computation tree logic (CTL) was introduced by Turing award winners Clarke and Emerson [2]. 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 [4], 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 φ `AU` φ 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 [3]. 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 [3, 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 [3, 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.

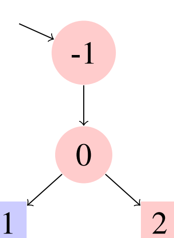
```
@Override
public Formula visitImplies(ImpliesContext context) {
  Formula right = (Formula) visit(context.formula(1));
  Formula left = (Formula) visit(context.formula(0));
  return new Implies(left, right);
}
```

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 have not been explored yet, that is, they are not part of $\rightarrow$.

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 $\forall\Box\,(\text{red} \vee \text{blue})$. 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 finite 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 or 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 complete 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 complete path if $s_i \to s_{i+1}$ for all $i \geq 0$.

Note that we require that the final state of a finite complete path is fully explored. We denote the set of complete paths that start in state s by $\mathit{CoPaths}(s)$.

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 partial paths that start in state s by $\mathit{PaPaths}(s)$.

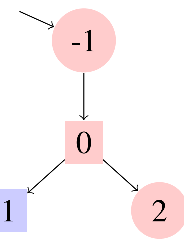
Definition 4. Let $(S, F, \to, I, AP, L)$ be a partial transition system.

- 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 $\mathit{PoPaths}(s)$. We denote the set of all paths that start in state s by $\mathit{Paths}(s)$, that is, $\mathit{Paths}(s) = \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) \cup \mathit{PoPaths}(s)$.

Consider the following partial transition system.



We have that

$$\begin{aligned}
CoPaths(-1) &= \{-1\ 0\ 2\} \\
PaPaths(-1) &= \{-1, -1\ 0, -1\ 0\ 1\} \\
PoPaths(-1) &= \{\, -1\ 0\ \pi \mid \pi[0] \notin \{1,2\} \wedge \pi \in S^\infty \,\} \cup \{\, -1\ 0\ 1\ \pi \mid \pi \in S^\infty \,\}
\end{aligned}$$

Since state 0 is not fully explored yet, we know that this state may have 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 $-1\ 0\ 2$. The sequence $-1\ 0\ 1$ is a partial path.

A partial transition system in which all states are fully explored, that is, an ordinary transition system, has no potential paths. Furthermore, each partial path can be extended to a complete path.

Proposition 1. *If $F = S$ then for all $s \in S$,*

1. *$PoPAths(s) = \emptyset$ and*

2. *for all $\pi \in PaPaths(s)$ there exists $\rho \in CoPaths(s)$ such that $|\rho| \geq |\pi|$ and for all $0 \leq i < |\pi|$, $\pi[i] = \rho[i]$.*

Proof.

1. Immediate from the definition of potential paths ($s_n \notin F$).

2. Each partial path can be extended to a complete path since every state is fully explored.

□

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, $[\![\forall \bigcirc \text{red}]\!](0) = \bot$ 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, $\top$, $\bot$ and $?$ by $\mathbb{V}$, that is,

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

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	$\bot$	$\top$	$\top$	$\top$
	$?$	$\top$	$?$	$?$

Equivalence is defined as follows.

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

We denote the length of a path π by $|\pi|$. If the 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.

- $[\![a]\!](s) = \begin{cases} \top & \text{if } a \in L(s) \\ \bot & \text{otherwise} \end{cases}$
- $[\![\texttt{true}]\!](s) = \top$
- $[\![\texttt{false}]\!](s) = \bot$
- $[\![\neg\varphi]\!](s) = \neg[\![\varphi]\!](s)$
- $[\![\varphi \wedge \psi]\!](s) = [\![\varphi]\!](s) \wedge [\![\psi]\!](s)$
- $[\![\varphi \vee \psi]\!](s) = [\![\varphi]\!](s) \vee [\![\psi]\!](s)$
- $[\![\varphi \rightarrow \psi]\!](s) = [\![\varphi]\!](s) \rightarrow [\![\psi]\!](s)$
- $[\![\varphi \leftrightarrow \psi]\!](s) = [\![\varphi]\!](s) \leftrightarrow [\![\psi]\!](s)$
- $[\![\forall \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \forall \pi \in \mathit{Paths}(s) : |\pi| > 1 \Rightarrow [\![\varphi]\!](\pi[1]) = \top \\ \bot & \text{if } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \bot \\ & \text{or } \mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \bot \\ ? & \text{otherwise} \end{cases}$
- $[\![\exists \bigcirc \varphi]\!](s) = \begin{cases} \top & \text{if } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \top \\ & \text{or } \mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \top \\ \bot & \text{if } \forall \pi \in \mathit{Paths}(s) : |\pi| > 1 \Rightarrow [\![\varphi]\!](\pi[1]) = \bot \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \Box \varphi]\!](s) = \begin{cases} \top & \text{if } \forall \pi \in \mathit{Paths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\ \bot & \text{if } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\ ? & \text{otherwise} \end{cases}$
- $[\![\exists \Box \varphi]\!](s) = \begin{cases} \top & \text{if } \exists \pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\ & \text{or } \mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\ \bot & \text{if } \forall \pi \in \mathit{CoPaths}(s) \cup \mathit{PoPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \Diamond \varphi]\!](s) = \begin{cases} \top & \text{if } \forall \pi \in \mathit{CoPaths}(s) \cup \mathit{PoPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\ \bot & \text{if } \exists \pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\ & \text{or } \mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\ ? & \text{otherwise} \end{cases}$

- $[\![\exists \Diamond \varphi]\!](s) = \begin{cases} \top & \text{if } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\ \bot & \text{if } \forall \pi \in \mathit{Paths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\ ? & \text{otherwise} \end{cases}$
- $[\![\forall \varphi \mathsf{U} \psi]\!](s) =$ details still need to be added here.
- $[\![\exists \varphi \mathsf{U} \psi]\!](s) =$ details still need to be added here.

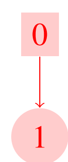
Which paths to consider for the formulas that start with a quantifier is subtle. Next, we discuss our choices and motivate them by means of examples.

$[\![\forall \bigcirc \varphi]\!](s) = \top$

To conclude that $[\![\forall \bigcirc \varphi]\!](s)$ is true, we need that all successors of state s satisfy φ. If state s is fully explored, we need to consider all successors:

$$\{\, \pi[1] \mid \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) \,\} = \{\, \pi[1] \mid \pi \in \mathit{Paths}(s) \,\}.$$

Consider the following partial transition system.

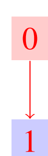


Since state 0 is not fully explored, state 0 has transitions to states other than state 1. Those other states may not be red and, hence, we do not know whether $\forall \bigcirc$ red holds in state 0. Therefore, for a non-fully explored state s, we consider also all potential successors of state s when determining if $\forall \bigcirc \varphi$ holds in s:

$$\{\, \pi[1] \mid \pi \in \mathit{Paths}(s) \,\}$$

$[\![\forall \bigcirc \varphi]\!](s) = \bot$

To conclude that $[\![\forall \bigcirc \varphi]\!](s)$ is false, we need to find a counterexample, that is, a successor of state s in which φ does not hold. Consider the following partial transition system.



Since state 0 transitions to a state that is not red, we can conclude that $\forall \bigcirc$ red does not hold in state 0. Because state 1 is not fully explored, it has unexplored outgoing transitions. Hence, the path 0 1 is not complete. Therefore, we consider both complete and partial paths starting from state s when determining if $[\![\forall \bigcirc \varphi]\!](s) = \bot$.

Consider the following partial transition system.

Since state 0 is not fully explored, it has transitions. As a consequence, the formula $\forall \bigcirc \texttt{false}$ does not holds in state 0. More generally, if there are potential paths starting of state s and the second state of those potential paths does not satisfy φ, then we can conclude that $\forall \bigcirc \varphi$ does not holds in s.

$[\![\exists \bigcirc \varphi]\!](s) = \top$

To conclude that $[\![\exists \bigcirc \varphi]\!](s)$ is true, we need a witness, that is, a successor of state s that satisfies φ. Consider the following partial transition system.

Since state 0 transitions to a state that is red, we can conclude that $\exists \bigcirc$ red holds in state 0. Because state 1 is not fully explored, it has unexplored outgoing transitions. Hence, the path $0\ 1$ is not complete. Therefore, we consider both complete and partial paths starting from state s when determining if $[\![\exists \bigcirc \varphi]\!](s) = \top$. Obviously, potential paths should not be considered.

Consider the following partial transition system.

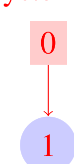
Since state 0 is not fully explored, it has transitions. As a consequence, the formula $\forall \bigcirc \texttt{true}$ holds in state 0. More generally, if there are potential paths starting of state s and the second state of those potential paths satifies φ, then we can conclude that $\forall \bigcirc \varphi$ holds in s.

$[\![\exists \bigcirc \varphi]\!](s) = \perp$

To conclude that $[\![\exists \bigcirc \varphi]\!](s)$ is false, we need that none of the successors of state s satisfy φ. If state s is fully explored, we need to consider all successors:

$$\{\, \pi[1] \mid \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) \,\} = \{\, \pi[1] \mid \pi \in \mathit{Paths}(s) \,\}.$$

Consider the following partial transition system.



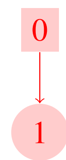
Since state 0 is not fully explored, state 0 may have transitions to states other than state 1. Those other states may be red and, hence, we do not know whether $\exists \bigcirc$ red does not hold in state 0. Therefore, for a non-fully explored state s, we consider all actual and potential successors of state s when determining if $\exists \bigcirc \varphi$ does not hold in s:

$$\{\, \pi[1] \mid \pi \in \mathit{Paths}(s) \,\}$$

$\llbracket \forall \Box \varphi \rrbracket (s) = \top$

To conclude that $\llbracket \forall \Box \varphi \rrbracket (s)$ is true, we need that all states reachable from state s satisfy φ.

Consider the following partial transition system.



States different from state 1 that are reachable from state 0 by transitions that have not been explored yet may not be red in which case $\forall\Box$red does not hold. Therefore, we consider all paths, including the potential paths, starting from state s when determining if $\llbracket \forall \Box \varphi \rrbracket (s) = \top$.

$\llbracket \forall \Box \varphi \rrbracket (s) = \bot$

To conclude that $\llbracket \forall \Box \varphi \rrbracket (s)$ is false, we need to find a counterexample, that is, a path starting in state s that contains a state that does not satisfy φ. We consider both complete and partial paths, but not potential paths.

$\llbracket \exists \Box \varphi \rrbracket (s) = \top$

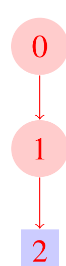
To conclude that $\llbracket \exists \Box \varphi \rrbracket (s)$ is true, we need a witness, that is, a complete path starting in state s of which all states satisfy φ.

Consider the following partial transition system.

Since state 0 is not fully explored, it has transitions. As a consequence, the formula $\exists\Box\texttt{true}$ holds in state 0. More generally, if there are potential paths starting of state s and all the states of those potential paths satify φ, then we can conclude that $\exists\Box\varphi$ holds in s.

$\llbracket \exists \Box \varphi \rrbracket (s) = \bot$

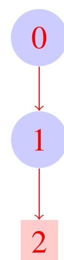
To conclude that $\llbracket \exists \Box \varphi \rrbracket (s)$ is false, we need that each path that starts in state s contains a state that does not satisfy φ. Consider the following partial transition system.



Since each path starting from state 0 contains state 2, which is blue, we can conclude that $\exists\Box$red does not hold in state 0. Note that we should consider both complete paths as well as potential paths. Partial paths should not be considered. For example, the partial path $0\ 1$ contains only red states.

$[\![\forall\Diamond\varphi]\!](s) = \top$

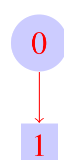
To conclude that $[\![\forall\Diamond\varphi]\!](s)$ is true, we need that each path contains a state that satisfies φ. Consider the following partial transition system.



Since each path starting from state 0 contains state 2, which is red, we can conclude that $\forall\Diamond$red holds in state 0. Note that we should consider both complete paths as well as potential paths. Partial paths should not be considered. For example, the partial path $0\ 1$ does not contain any red states.

$[\![\forall\Diamond\varphi]\!](s) = \bot$

To conclude that $[\![\forall\Diamond\varphi]\!](s)$ is false, we need to find a counterexample, that is, a path starting in state s of which all states do not satisfy φ. Consider the following partial transition system.



Since state 1 is not fully explored, the path $0\ 1$ is partial. This partial path may be the prefix of a complete path that includes red states and, hence, we cannot conclude that $[\![\forall\Diamond\text{red}]\!](0) = \bot$. Therefore, we consider complete paths only when determining if $[\![\forall\Diamond\varphi]\!](s) = \bot$.

Consider the following partial transition system.

Since state 0 is not fully explored, it has transitions. As a consequence, the formula $\forall\Diamond\texttt{false}$ does not hold in state 0. More generally, if there are potential paths starting of state s and all the states of those potential paths do not satify φ, then we can conclude that $\forall\Diamond\varphi$ does not hold in s.

$[\![\exists\Diamond\varphi]\!](s) = \top$

To conclude that $[\![\exists\Diamond\varphi]\!](s)$ is true, we need to find a witness, that is, a path starting in state s that contains a state that satisfies φ. The path can either be complete or partial.

$[\![\exists\Diamond\varphi]\!](s) = \bot$

To conclude that $[\![\exists\Diamond\varphi]\!](s)$ is false, no path starting in state s should contain a state that satisfies φ. Hence, we need to consider complete, partial, and potential paths.

7 Existential Normal Form

In [1, Definition 6.13], an existential normal form has been introduced. It restricts the syntax of CTL. In particular, it restricts to existential quantifiers, eliminating the universal quantifiers. Furthermore, in [1, Theorem 6.14], it is shown that for each CTL formula there exists an equivalent CTL formula in existential normal. Below, we extend these results to the setting of partial transition systems.

Definition 6. Let AP be the set of atomic propositions. The set of CTL formulas in existential normal form is defined by the following grammar.

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists \bigcirc \varphi \mid \exists\Box\varphi \mid \exists\Diamond\varphi \mid \exists\varphi \,\mathsf{U}\, \varphi \mid \forall\varphi \,\mathsf{U}\, \varphi$$

where $a \in AP$.

Next, we define equivalence of CTL formulas.

Definition 7. For CTL formulas φ and ψ,

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

Proposition 2. *Let $a \in AP$. For CTL formulas φ and ψ,*

1. $\texttt{true} \equiv a \vee \neg a$
2. $\texttt{false} \equiv a \wedge \neg a$
3. $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$
4. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
5. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
6. $\forall \bigcirc \varphi \equiv \neg\exists \bigcirc \neg\varphi$
7. $\forall\Box\varphi \equiv \neg\exists\Diamond\neg\varphi$

8. $\forall\Diamond\varphi \equiv \neg\exists\Box\neg\varphi$

Proof. Let $(S, F, \rightarrow, I, AP, L)$ be a partial transition system and let $s \in S$.

1. If $a \in L(s)$, then

$$[\![\texttt{true}]\!](s) = \top = \top \vee \neg\top = [\![a]\!](s) \vee \neg[\![a]\!](s) = [\![a \vee \neg a]\!](s).$$

Otherwise,

$$[\![\texttt{true}]\!](s) = \top = \perp \vee \neg \perp = [\![a]\!](s) \vee \neg[\![a]\!](s) = [\![a \vee \neg a]\!](s).$$

2. If $a \in L(s)$, then

$$[\![\texttt{false}]\!](s) = \perp = \top \wedge \neg\top = [\![a]\!](s) \wedge \neg[\![a]\!](s) = [\![a \wedge \neg a]\!](s).$$

Otherwise,

$$[\![\texttt{false}]\!](s) = \perp = \perp \wedge \neg \perp = [\![a]\!](s) \wedge \neg[\![a]\!](s) = [\![a \wedge \neg a]\!](s).$$

3. It suffices to show that $v \vee w = \neg(\neg v \wedge \neg w)$ for all $v, w \in \mathbb{V}$.

v	w	$\neg v$	$\neg w$	$\neg v \wedge \neg w$	$\neg(\neg v \wedge \neg w)$
$\top$	$\top$	$\perp$	$\perp$	$\perp$	$\top$
$\perp$	$\top$	$\top$	$\perp$	$\perp$	$\top$
$?$	$\top$	$?$	$\perp$	$\perp$	$\top$
$\top$	$\perp$	$\perp$	$\top$	$\perp$	$\top$
$\perp$	$\perp$	$\top$	$\top$	$\top$	$\perp$
$?$	$\perp$	$?$	$\top$	$?$	$?$
$\top$	$?$	$\perp$	$?$	$\perp$	$\top$
$\perp$	$?$	$\top$	$?$	$?$	$?$
$?$	$?$	$?$	$?$	$?$	$?$

4. It suffices to show that $v \rightarrow w = \neg v \vee w$ for all $v, w \in \mathbb{V}$.

v	w	$\neg v$	$\neg v \vee w$
$\top$	$\top$	$\perp$	$\top$
$\perp$	$\top$	$\top$	$\top$
$?$	$\top$	$?$	$\top$
$\top$	$\perp$	$\perp$	$\perp$
$\perp$	$\perp$	$\top$	$\top$
$?$	$\perp$	$?$	$?$
$\top$	$?$	$\perp$	$?$
$\perp$	$?$	$\top$	$\top$
$?$	$?$	$?$	$?$

5. It suffices to show that $v \leftrightarrow w = (v \to w) \wedge (w \to v)$ for all $v, w \in \mathbb{V}$.

v	w	$v \to w$	$w \to v$	$(v \to w) \wedge (w \to v)$
$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\bot$	$\bot$
$?$	$\top$	$\top$	$?$	$?$
$\top$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\top$	$\top$
$?$	$\bot$	$?$	$\top$	$?$
$\top$	$?$	$?$	$\top$	$?$
$\bot$	$?$	$\top$	$?$	$?$
$?$	$?$	$?$	$?$	$?$

6. Since

$$\begin{aligned}
[\![\forall \bigcirc \varphi]\!](s) = \top &\text{ iff } \forall \pi \in \mathit{Paths}(s) : |\pi| > 1 \Rightarrow [\![\varphi]\!](\pi[1]) = \top \\
&\text{ iff } \forall \pi \in \mathit{Paths}(s) : |\pi| > 1 \Rightarrow [\![\neg\varphi]\!](\pi[1]) = \bot \\
&\text{ iff } [\![\exists \bigcirc \neg\varphi]\!](s) = \bot \\
&\text{ iff } [\![\neg\exists \bigcirc \neg\varphi]\!](s) = \top
\end{aligned}$$

and

$$\begin{aligned}
[\![\forall \bigcirc \varphi]\!](s) = \bot &\text{ iff } (\exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \bot) \vee \\
&\qquad (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in (\mathit{PoPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \bot) \\
&\text{ iff } (\exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\neg\varphi]\!](\pi[1]) = \top) \vee \\
&\qquad (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in (\mathit{PoPaths}(s) : |\pi| > 1 \wedge [\![\neg\varphi]\!](\pi[1]) = \top) \\
&\text{ iff } [\![\exists \bigcirc \neg\varphi]\!](s) = \top \\
&\text{ iff } [\![\neg\exists \bigcirc \neg\varphi]\!](s) = \bot
\end{aligned}$$

we can conclude that $\forall \bigcirc \varphi \equiv \neg\exists \bigcirc \neg\varphi$.

7. Since

$$\begin{aligned}
[\![\forall\Box\varphi]\!](s) = \top &\text{ iff } \forall \pi \in \mathit{Paths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[1]) = \top \\
&\text{ iff } \forall \pi \in \mathit{Paths}(s) : \forall 0 \leq i < |\pi| : [\![\neg\varphi]\!](\pi[1]) = \bot \\
&\text{ iff } [\![\exists\Diamond\neg\varphi]\!](s) = \bot \\
&\text{ iff } [\![\neg\exists\Diamond\neg\varphi]\!](s) = \top
\end{aligned}$$

and

$$\begin{aligned}
[\![\forall\Box\varphi]\!](s) = \bot &\text{ iff } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[1]) = \bot \\
&\text{ iff } \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : [\![\neg\varphi]\!](\pi[1]) = \top \\
&\text{ iff } [\![\exists\Diamond\neg\varphi]\!](s) = \top \\
&\text{ iff } [\![\neg\exists\Diamond\neg\varphi]\!](s) = \bot
\end{aligned}$$

we can conclude that $\forall\Box\varphi \equiv \neg\exists\Diamond\neg\varphi$.

8. Since

$$\begin{aligned}
[\![\forall\Diamond\varphi]\!](s) = \top \text{ iff } & \forall\pi \in \mathit{CoPaths}(s) \cup \mathit{PoPaths}(s) : \exists 0 \le i < |\pi| : [\![\varphi]\!](\pi[1]) = \top \\
\text{iff } & \forall\pi \in \mathit{CoPaths}(s) \cup \mathit{PoPaths}(s) : \exists 0 \le i < |\pi| : [\![\neg\varphi]\!](\pi[1]) = \bot \\
\text{iff } & [\![\exists\Box\neg\varphi]\!](s) = \bot \\
\text{iff } & [\![\neg\exists\Box\neg\varphi]\!](s) = \top
\end{aligned}$$

and

$$\begin{aligned}
[\![\forall\Diamond\varphi]\!](s) = \bot \;\; \text{iff } & (\exists\pi \in \mathit{CoPaths}(s) : \forall 0 \le i < |\pi| : [\![\varphi]\!](\pi[1]) = \bot) \vee \\
& \quad (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall\pi \in \mathit{PoPaths}(s) : \forall 0 \le i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot) \\
\text{iff } & (\exists\pi \in \mathit{CoPaths}(s) : \forall 0 \le i < |\pi| : [\![\neg\varphi]\!](\pi[1]) = \top) \vee \\
& \quad (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall\pi \in \mathit{PoPaths}(s) : \forall 0 \le i < |\pi| : [\![\neg\varphi]\!](\pi[i]) = \top) \\
\text{iff } & [\![\exists\Box\neg\varphi]\!](s) = \top \\
\text{iff } & [\![\neg\exists\Box\neg\varphi]\!](s) = \bot
\end{aligned}$$

we can conclude that $\forall\Diamond\varphi \equiv \neg\exists\Box\neg\varphi$.

□

Corollary 1. *For each CTL formula there exists an equivalent CTL formula in existential normal form.*

8

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 (along the lines of [**?**]) as follows:

- $s \models \exists \bigcirc \varphi$ iff $\exists\pi \in \mathit{CoPaths}(s) : |\pi| > 1 \wedge \pi[1] \models \varphi$
- $s \models \exists\Box\varphi$ iff $\exists\pi \in \mathit{CoPaths}(s) : \forall 0 \le i < |\pi| : \pi[i] \models \varphi$
- $s \models \exists\Diamond\varphi$ iff $\exists\pi \in \mathit{CoPaths}(s) : \exists 0 \le i < |\pi| : \pi[i] \models \varphi$
- $s \models \exists\varphi \,\mathsf{U}\, \psi$ iff $\exists\pi \in \mathit{CoPaths}(s) : \exists 0 \le j < |\pi| : \pi[j] \models \psi \wedge \forall 0 \le k < j : \pi[k] \models \varphi$
- $s \models \forall\varphi \,\mathsf{U}\, \psi$ iff $\forall\pi \in \mathit{CoPaths}(s) : \exists 0 \le j < |\pi| : \pi[j] \models \psi \wedge \forall 0 \le k < j : \pi[k] \models \varphi$

Proposition 3. *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

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

  and

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

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

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

  and

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

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

$$
\begin{aligned}
& [\![\exists \bigcirc \varphi]\!](s) = \top \\
\text{iff } & (\exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \top) \vee \\
& (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \top) \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge [\![\varphi]\!](\pi[1]) = \top \qquad \text{[Proposition 1.1]} \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \wedge \pi[1] \models \varphi \qquad \text{[by induction]} \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) : |\pi| > 1 \wedge \pi[1] \models \varphi \qquad \text{[Proposition 1.2]} \\
\text{iff } & s \models \exists \bigcirc \varphi
\end{aligned}
$$

and

$$
\begin{aligned}
& [\![\exists \bigcirc \varphi]\!](s) = \bot \\
\text{iff } & \forall \pi \in \mathit{Paths}(s) : |\pi| > 1 \Rightarrow [\![\varphi]\!](\pi[1]) = \bot \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \Rightarrow [\![\varphi]\!](\pi[1]) = \bot \qquad \text{[Proposition 1.1]} \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : |\pi| > 1 \Rightarrow \pi[1] \not\models \varphi \qquad \text{[by induction]} \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) : |\pi| > 1 \Rightarrow \pi[1] \not\models \varphi \qquad \text{[Proposition 1.2]} \\
\text{iff } & s \not\models \exists \bigcirc \varphi
\end{aligned}
$$

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

$$
\begin{aligned}
& [\![\exists \Box \varphi]\!](s) = \top \\
\text{iff } & (\exists \pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top) \vee \\
& (\mathit{PoPaths}(s) \neq \emptyset \wedge \forall \pi \in \mathit{PoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top) \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \qquad \text{[Proposition 1.1]} \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : \pi[i] \models \varphi \qquad \text{[by induction]} \\
\text{iff } & s \models \exists \Box \varphi
\end{aligned}
$$

and

$$
\begin{aligned}
& [\![\exists \Box \varphi]\!](s) = \bot \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) \cup \mathit{PoPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \qquad \text{[Proposition 1.1]} \\
\text{iff } & \forall \pi \in \mathit{CoPaths}(s) : \exists 0 \leq i < |\pi| : \pi[i] \not\models \varphi \qquad \text{[by induction]} \\
\text{iff } & s \not\models \exists \Box \varphi
\end{aligned}
$$

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

$$
\begin{aligned}
& [\![\exists \Diamond \varphi]\!](s) = \top \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \top \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \exists 0 \leq i < |\pi| : \pi[i] \models \varphi \qquad \text{[by induction]} \\
\text{iff } & \exists \pi \in \mathit{CoPaths}(s) : \exists 0 \leq i < |\pi| : \pi[i] \models \varphi \qquad \text{[Proposition 1.2]} \\
\text{iff } & s \models \exists \Diamond \varphi
\end{aligned}
$$

and

$$\begin{aligned}
& [\![\exists\Diamond\varphi]\!](s) = \bot \\
\text{iff } & \forall\pi \in \mathit{Paths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \\
\text{iff } & \forall\pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \forall 0 \leq i < |\pi| : [\![\varphi]\!](\pi[i]) = \bot \qquad \text{[Proposition 1.1]} \\
\text{iff } & \forall\pi \in \mathit{CoPaths}(s) \cup \mathit{PaPaths}(s) : \forall 0 \leq i < |\pi| : \pi[i] \not\models \varphi \qquad \text{[by induction]} \\
\text{iff } & \forall\pi \in \mathit{CoPaths}(s) : \forall 0 \leq i < |\pi| : \pi[i] \not\models \varphi \qquad \text{[Proposition 1.2]} \\
\text{iff } & s \not\models \exists\Diamond\varphi
\end{aligned}$$

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

□

For each CTL formula we define the following two sets of states.

Definition 8. Let φ be a CTL formula. Then

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

References

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

[2] 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.

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

[4] 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.

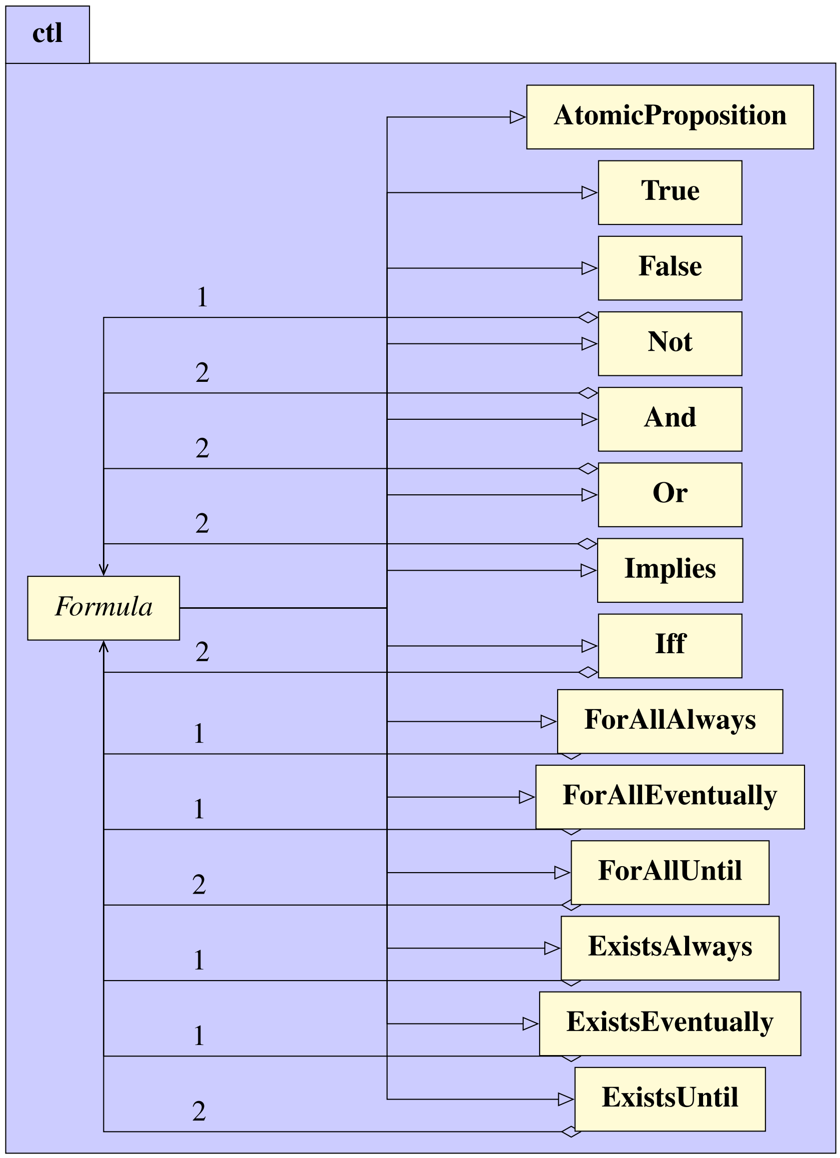


Figure 1: UML class diagram of the abstract syntax classes.