# Logic Systems Overview: Modal Logic, Computation Tree Logic, Probabilistic Temporal Logic

D. Gueorguiev 9/9/2023

### Review on Computation Tree Logic and Specification Language in (E.M. Clarke, 1983)

In this Appendix section we will review an efficient procedure for verifying that a finite state concurrent system meets a specification expressed in a (propositional) branching-time temporal logic. The reviewed algorithm has linear complexity in both the size of the specification and the size of the global transition. The global state graph can be viewed as a finite *Kripke* structure, and an efficient algorithm can be given to determine whether a given structure is a model of a particular formula. The algorithm, which we call a *model checker*, is similar to the global flow analysis algorithms used in compiler optimization and has complexity linear in both the size of the structure and the size of the specification.

#### The Specification Language

The syntax for CTL is given below. is the underlying set of *atomic propositions*.

1. Every atomic proposition is a CTL formula
2. If and are CTL formulae, the so are , , , , , .

The symbols and have their usual meanings. is the *nexttime* operator; the formulae () intuitively means that holds in every (in some) immediate successor of the current program state. is the *until* operator; The formula () intuitively means that for every computation path (for some computation path), there exists an initial prefix of the path such that holds at the last state of the prefix and holds at all other states along the prefix.

The semantics of CTL formulae with respect to a labeled state-transition graph is defined below. Formally, a CTL structure is a triple where

1. is a finite set of states
2. is a binary relation on i.e., . It gives the possible transitions between states and must be total i.e., .
3. is an assignment of atomic propositions to states i.e.,

A *path* is an infinite sequence of states () such that . For any structure and state , there is an *infinite computation tree* with root labeled such that is an arc in the tree *iff* .

We use the standard notation to indicate truth in a structure: means that formula holds at state in structure . When the structure is understood, we simply write . The relation is defined inductively as follows:

iff

iff not

iff and

iff for all states t such that ,

iff for all states t such that ,

iff for all paths () s.t.

iff for some path () s.t.

#### Model Checker

Assume that we wish to determine whether formula is true in the finite structure . When the algorithm finishes, each state will be labelled with the set of subformulae true in the state. We let denote this set for state . Consequently, iff at termination. We first consider the case in which each state is currently labelled with the *immediate* subformulae of which are true in that state. We will use the following primitives for manipulating formulas and accessing the labels associated with states:

* and give the first and second arguments of a two-argument formula such as
* will return true (false) if state is (is not) labelled with formula .
* adds formula f to the current label of state .

The state labeling algorithm (procedure ) must be able to handle seven cases depending on whether f is atomic or has one of the following forms: ,,. We will only consider the case in which here since all other cases are either straightforward or similar. For the case the algorithm uses depth first search to explore the state graph. The bit array is used to indicate which states have been visited by the search algorithm. The algorithm also uses stack ST to keep track of those states which require additional processing before the truth or falsity of can be determined. The Boolean procedure will determine (in constant true) whether state is currently on the stack .

def label\_graph(f, b):

“””

f: formula

b (bool): result

“””

ST = empty\_stack

for s in S:

marked(s) = false

for s in S:

if not marked(s):

au(f, s, b)

def au(f, s, b):

“””

f: formula

s: state

b (bool): result

“””

“””

If s is marked and stacked, return false (see lemma 3.1).

If s is already labelled with f, then return true. Otherwise,

If s is marked but nether stacked nor labelled, then return false

if marked(s):

if stacked(s):

b = False

return

if labelled

//Appendix: Finish the paragraph on CTL algorithms

### Definitions and Review on Probabilistic Real Time Computation Tree Logic (PCTL) in (Hansson & Jonsson, 1994)

#### Notation

Assume a finite set of *atomic propositions*. We use , , etc. to denote atomic propositions. Formulas in PCTL are built from atomic propositions, propositional logic connectives and operators for expressing time and probabilities. The set of PCTL formulas is divided into *path formulas* and *state formulas*. Their syntax is defined inductively as follows:

* Each atomic proposition is a state formula
* If and are state formulas, then so are , , ,
* If and are state formulas, and is a nonnegative integer or , then and are path formulas,
* If is a path formula and is a real number with , then and are state formulas.

We shall use , , etc. to range over PCTL formulas. Intuitively, state formulas represent properties of states and path formulas represent properties of paths (i.e., sequences of states). The propositional connectives , , , and have their usual meanings. The operator is the *(strong) until* operator, and is the *unless* (or *weak until*) operator. For a given state , the formulas and express that holds for a path from with a probability of at least and greater than , respectively.

We shall use as a shorthand for , and use as a shorthand for . Intuitively, means that there is at least a probability that both will become true within time units and that will be true from now on until becomes true. means that there is at least a probability that either will remain true for at least time units, or that both will become true within time units and that will be true from now on until becomes true.

PCTL formulas are interpreted over structures that are discrete time Markov chains. A specified initial state is associated with the structure. In addition, for each state there is an assignment of truth values to atomic propositions appearing in a given formula. Formally, a structure is a quadruple , where

is a finite set of states, ranged over by , , etc.,

is an *initial state*,

is a *transition probability function*, , such that for all in we have

,

is a labeling function assigning atomic propositions to states, i.e.,

Intuitively, each transition is considered to require one *time unit*. We will display structures as transition diagrams, where states (circles) are labeled with atomic propositions and transitions with non-zero probability are represented as arrows labeled with their probabilities (e.g., the arrow going from state to state is labelled with ). The initial state is indicated with an extra arrow.

A path from a state in a structure is an infinite sequence

of states with as the first state. The :th state () of is denoted , and the prefix of of length is denoted , i.e.,

For each structure and state we define a probability measure on the set of paths from . is defined on the probability space , where is the set of paths starting in and is a sigma-algebra on generated by sets

Of paths with a common finite prefix . The measure is defined as follows: for each finite sequence of states,

i.e., the measure of the set of paths for which is equal to the product . For we define . This uniquely defines the measure on all sets of paths in the sigma-algebra .

We define the truth of PCTL formulas for a structure by a satisfaction relation:

which means that the state formula is true at state in the structure . In order to define the satisfaction relation for states, it is helpful to use another relation

which means that the path satisfies the path formula in . The relations and are inductively defined as follows:

iff

iff not

iff and

iff or

iff or

iff there exist an such that and

iff or

iff the -measure of the set of paths starting in for which is at least .

iff the -measure of the set of paths starting in for which is greater than .

We define

where is the initial state of .

#### Properties expressible in PCTL

We will present examples of properties that can be expressed in PCTL. First, we discuss some of the facilities of PCTL which makes it suitable for specification of soft and hard deadlines.

The main difference between PCTL and branching time temporal logics such as CTL, is the quantification over paths and the ability to specify quantitative time. CTL allows universal () and existential () quantification over paths, i.e., one can state that a property should hold for all computations (paths) or that it should hold for some computations (paths). It is not possible to state that a property should hold for a certain portion of the computations, e.g., for at least 50% of the computations. In PCTL, on the other hand, arbitrary probabilities can be assigned to path formulas, thus obtaining a more general quantification over paths. An analogy to universal and existential quantification can in PCTL be defined as:

Quantitative time allows us to specify time-critical properties that relate the occurrence of events of a system in real-time. In PCTL it is possible to state that a property will hold continuously during a specific time interval, or that a property will hold sometime during a time interval. Combining this with the above quantification we can define

means that the formula holds continuously for time units with a probability of at least , and means that the formula holds within time units with a probability of at least .

An important requirement on most real-time and distributed systems is that they should be continuously operating, e.g., every time the controller receives an alarm signal from a sensor the controller should take the appropriate action. We can express such requirements with the following PCTL operators:

means that is always (in all states that can be reached with non-zero probability), means that a state where is will eventually be reached with probability 1, means that there is a non-zero probability for to be continuously true, and means that there exists a state where holds which can be reached with non-zero probability.

**Definition** (Susan Owicki, 1982)[[1]](#footnote-1): (unquantified) *leads-to* operator ()

Whenever a becomes true, b will eventually hold.

**Definition**: PCTL quantified *leads-to* operator ():

means that whenever holds there is a probability of at least p that will hold within time units.

Many modal operators can be derived from the basic PCTL operators. We can for instance define an operator that corresponds to the CTL operator (E.M. Clarke, 1983) as follows:

As an example, we will specify a mutual exclusion property. Consider two processes ( and ) using the same criticial section. The atomic propositions , , and indicates that is in its non-critical, trying, and critical regions, respectively. The mutual exclusion property can be expressed as:

This is not sufficient for most *real-time systems* since the property only states that simultaneous access to the critical section must be avoided always under all circumstances. To capture a specific real-time behavior, we can specify that whenever enters its trying region, it will enter its critical region within 4 time units. This can in PCTL be expressed as:

For some systems, it might be sufficient that the deadline is almost always met (e.g. in 99% of the cases). The relaxed property can be expressed as:

Relaxing the timing requirement might enable a less costly implementation that still shows acceptable behavior. To be on the safe side we could add a strict upper limit to the relaxed property, combining the hard and soft deadlines above. If we assume that we want to always enter its critical region within 10 time units, and almost always within 4 time units we get the property:

#### Model Checking in PCTL

In this section we present a model checking algorithm, which given a structure and a PCTL formula determines whether . The algorithm is based on the algorithm for model checking CTL (E.M. Clarke, 1983). It is designed so that when it finishes each state will be labeled with the set of subformulas of that are in the state. One can then conclude that if the initial state () is labeled with .

For each state of the structure, the algorithm uses a variable to indicate the subformulas that are in state . Initially, each state is labeled with the atomic propositions that are in , i.e., . The labeling is then performed starting with the smallest subformula of that has not yet been labeled and ending with labeling states with itself. Composite formulas are labeled based on the labeling of their parts. Assuming that we have performed the labeling of and , the labeling corresponding to negation () and propositional connectives () is straightforward, i.e.

if ,

if ,

if or

if or

where in addition the new formula must be a subformula of . The next section presents two algorithms for labeling states with the modal subformulas of PCTL. After that, in the subsequent section, we discuss labeling in cases with extreme parameter values (e.g., , , and ).

#### Labeling states with the modal subformulas of PCTL

We shall give an algorithm for the labeling of states for the formula assuming that we have done the labeling for formulas and , and that .

//Appendix: Finish the paragraph on PTL Theory

### Definitions and Review on Probabilistic Temporal Logic in (Kleinberg, Causality, Probability, and Time, 2012)

*Probabilistic Temporal Logic* (PTL) is a tool for state machine model checking which is a more complete alternative of the Labeled DFG defined earlier. A somewhat reduced subset of Probabilistic Temporal Logic is defined with the help of *Kripke* structures. With randomness introduced *Kripke* structure is roughly equivalent to a Discrete Time Markov Chain, and it is just another tool to validate specific first order logic statements relevant for RCA against our process model.

**Definition B1**: *Kripke structure*

Let be a set of atomic propositions. A *Kripke* structure over is defined as the tuple where

* is a finite set of states
* is the set of initial states
* is a total transition relation, such that
* : is a function which labels each state with a set of atomic propositions that are true within it.

The function (relation) being a total transition function (relation) means that for every state, there is at least one transition from that state (to itself or to another state). The function (relation) maps states to the truth values of propositions at that state. Since there are propositions, there are possible truth values and maps each state to one of these.

A *path* in a *Kripke* structure is an infinite sequence of states. Precisely, a path is a sequence of states () such that for every , . That says that the series of transitions described in the sequence is possible. The notation is used to denote the *subpath suffix* of the path starting with state .

To find the properties that are true in such kind of structures we need a formal method for representing the properties to be tested. There are number of temporal logic systems which express (slightly) different sets of formula.

We are going to introduce *Computational Tree Logic* (CTL) system which will be used to build upon later and define PTL.

The formulas in CTL are composed of paired *path quantifiers* and *temporal operators*. Path quantifiers describe whether a property holds ***for all paths*** (denoted with the operator ), or ***for some path*** (denoted with operator ), starting at a given state. The temporal operators describe where along the path the properties will hold. For example, if is some state, then is a valid CTL formula, but is not, since is not paired with one of or . More formally,

* *Finally* () – at some state on the path the property will hold
* *Globally* (G) – the property will hold along the entire path
* *Next* () – the property will hold at the next state of the path
* *Until* () – applies to two properties, the first one holds in every state along the path until at some state the second property holds
* *Weak Until aka Until or Release* (W)

//Finish this paragraph on CTL

As in CTL, in PTL there are two types of formulas: *path formulas* and *state formulas*. State formulas express properties that must hold within a state, such as being labeled with certain atomic propositions, while path formulas refer to sequences of states along which the formula must hold. The formulas are comprised of atomic propositions , propositional logical connectives (such as ), and the modal operators denoting time and probability. The logic syntax tells how valid PTL formulas are constructed:

1. All atomic propositions are state formulas
2. If and are state formulas, so are , , , and

### Examples of PTL

: Event is reachable from event with probability at least p after at least r steps and at most s steps

//Finish the paragraph on PTL Examples

**Definition B2**: *prima facie* cause expressed with PTL formulas

These conditions mean that 1) a state where is true will be reached with non-zero probability and 2) the probability of reaching a state where e is true (within the time bounds) is greater after being in a state where c is true (probability ) than 3) it is by simply starting from initial state of the system (probability ). When making inferences from data that means that must occur at least once, and the conditional probability of given is greater than the marginal probability of (usually calculated from frequencies). Since negative (probability lowering) causes can be defined in terms of their complement (so that if lowers the probability of , raises its probability, the definition here is in terms of positive, probability raising causes.

#### Equivalence between the causality concepts based on PTL and Suppes’ causal framework

**Theorem B3**: Assume there is a *Kripke* structure representing the underlying system governing the occurrences of the events. Then the conditions for causality given in the **Definition B2** for prima facie cause earlier are satisfied if and only if the conditions for causality given by **Definition A3** are satisfied.

*Proof:*

We begin by showing that **Definition B2Definition A3** and then show that **Definition A3Definition B2**.

**Proposition B1.1**: **Definition B2Definition A3**

*Proof:*

Assume that , and there is a *Kripke* structure , representing the underlying system governing the occurrences of these events. Also assume that states in that satisfy and are labeled as such. If in **Definition A3**, we assume that in that satisfy and are labeled as such. If in **Definition A3**, we assume that in there will be at least one transition between an event at and one at . That is, the timescale of is as fine as that of Suppes and vice versa. Further, we assume that the probabilities of Suppes’s formulation and those in come from the same source and this if represented correctly, in **Definition A3** is equal to in **Definition B2**.

*Condition 1*:

By definition of , the probability of occurring at any time is less than . Recall that the probability of a path is the product of the transition probabilities along the path, and the probability of a set of paths is the sum of their individual path probabilities. For a structure to satisfy this formula, the set of paths from the start state that reach a state where holds must be less than , and the probability of reaching a state where holds in this system is less than . Thus,

(A.1)

Now we must show . We now show that this conditional probability is greater than or equal to if:

(A.2)

is satisfied.

The probability of a transition from state to state that labels the edge between them,

,

Is the conditional probability:

(A.3)

The probability of reaching one time unit after . Then, for a path:

,

we can calculate the probability, given , of reaching (via ) within two time units:

(A.4)

and since and are independent conditioned on this becomes:

(A.5)

Note that the probabilities of the righthand side are simply the transition probabilities from to , and to (since there is one time unit between the states, they can only be reached via single transition).

Thus, the conditional probability is precisely the path probability:

(A.6)

Then, if we have a set of paths from to , the conditional probability is the sum of these path probabilities. For example, we may have the following paths:

In which case:

(A.7)

and from eq. (A.6) this becomes:

(A.8)

the sum of the individual path probabilities. Let us now assume that is labeled with and is labeled with , these are the only and states in the system, and there are no other paths between the states taking less than or equal to 2 time units. Then, this probability we have computed is in fact the probability of:

(A.9)

since the probability of reaching , during a window of time simply means looking at the set of paths reaching during that window. Similarly, to find the probability of:

(A.10)

we must consider the set of paths from states labeled with to those labeled with that take at least 1 time unit. Since there can be cycles in our graph, calculating the probability associated with a leads-to formula with an infinite upper time bound requires a slightly different method.

//Finish the paragraph the leads-to formula with lower and upper bound

*Leads-to with Both Lower and Upper Time Bounds*

This paragraph deals with evaluation of *Leads-To* with applied window of time in which leads to . We assume a minimum time after is true before which is true. Here it is shown that it is possible to add such a lower bound. By Definition:

(A.11)

where . Thus, we are only adding a minimum time to the consequent of the conditional. If we can label states where is true, then we can proceed as in the algorithm of (Hansson & Jonsson, 1994).

//Finish the paragraph on Leads-to with Both Lower and Upper Bounds taken from the B.2 of (Kleinberg, Causality, Probability, and Time, 2012)

# Bibliography

Clarke, E. M., & Schlingloff, B. H. (2001). Model Checking. In A. Robinson, & A. Voronkov, *Handbook of Automated Reasoning* (pp. 1369-1520). Elsevier Science Publishers B.V.

E.M. Clarke, E. E. (1983). Automatic Verification Of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach. *ACM*, 117-126.

Eells, E. (1991). *Probabilistic Causality.* Cambridge UK: Cambridge University Press.

G.E. Hughes, M. C. (1996). *A New Introduction To Modal Logic.* London: Routledge.

Good, I. J. (1961). A Causal Calculus (I and II). *The British Journal for the Philosophy of Science*, 305-318, 43-51.

Hansson, H., & Jonsson, B. (1994). *A Logic for Reasoning about Time and Reliability.* Uppsala, Sweden: SICS Research Report SICS/R90013.

Houdth, G. V., Depaire, B., & Martin, N. (2022). Root Cause Analysis in Process Mining with Probabilistic Temporal Logic. *ICPM 2021 Workshops* (pp. pp. 73–84). Eindhoven, The Netherlands: LNBIP Volume 433.

Hume, D. (1748). *Philosophical Essays Concerning Human Understanding.* London: Printed for A. Millar, opposite Katharine-Street in the Strand. MDCCXLVII.

Kleinberg, S. (2010). *An Algorithmic Enquiry Concerning Causality.* New York: New York University.

Kleinberg, S. (2012). *Causality, Probability, and Time.* Cambridge, UK: Cambridge University Press.

Kleinberg, S., & Mishra, B. (2009). The Temporal Logic Of Causal Structures. *The Conference on Uncertainty in Artificial Intelligence*, (pp. 303-312). Montreal, Canada.

Lewis, D. (1973). *Counterfactuals.* Malden, Massachusetts: Blackwell Publishers.

Lewis, D. (1974). Causation. *Journal of Philosophy*, 556-567.

Mackie, J. (1980). *The Cement of The Universe: A Study of Causation.* New York: Oxford University Press, New York, United States.

Otte, R. E. (1982). *Probability and Causality, PhD Thesis.* Ann Arbor , MI, 48106: University of Arizona Graduate College, University Microfilm International.

Reichenbach, H. (1956). *The Direction of Time.* Berkeley and Los Angeles, California: University of California Press.

Salmon, W. C. (1980). Probabilistic Causality. *Pacific Philosophical Quarterly*, pp. 137-153.

Salmon, W. C. (1998). *Causality and Explanation.* Pittsburgh, Pennsylvania: Oxford University Press.

Spirtes, P., Glymour, C., & Sheines, R. (1993). *Causation, Prediction and Search.* New York: Springer Verlag.

Susan Owicki, L. L. (1982). Proving Liveness Properties of Concurrent Programs. *ACM Transactions on Programming Languages and Systems*, 455-495.

# Downloadable Links for the Bibliography

(Hume, Philosophical Essays Concerning Human Understanding, 1748): [here](https://en.wikisource.org/wiki/Philosophical_Essays_Concerning_Human_Understanding)

(Clarke & Schlingloff, 2001): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/ModelChecking_ClarkeSchlingloff1999.pdf)

(Eells, 1991): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/eells_probabilistic_causality_1991.pdf)

(Hansson & Jonsson, 1994): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/ALogicforReasoningaboutTimeandReliability_Hansson_Johnson_1994.pdf)

(Houdth, Depaire, & Martin, 2022): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/RootCauseAnalysisinProcessMiningwithProbabilisticTemporalLogicHoudt2022.pdf)

(Kleinberg & Mishra, The Temporal Logic Of Causal Structures, 2009): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/TheTemporalLogicofCausalStructures_Kleinberg_Mishra_2009.pdf)

(Kleinberg, An Algorithmic Enquiry Concerning Causality, 2010): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/Algorithmic_Enquiry_Concerning_Causality_Kleinberg_PhD_Thesis_2010.pdf)

(Reichenbach, 1956): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/the-direction-of-time-hans-reichenbach-ucal-press-1971.pdf)

(Spirtes, Glymour, & Sheines, 1993): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/CausationPredictionandSearch_Spirtes_CMU_2000.pdf)

(Otte, 1982): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/Probability_and_Causality_PhD_Thesis_Otte_1982.pdf)

(G.E. Hughes, 1996): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/huges_cresswell_modal_logic.pdf)

(E.M. Clarke, 1983): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/AutomaticVerifictionOfFiniteStateConcurrentSystemUsingTemporalLogicSpecification.pdf)

(Salmon, Causality and Explanation, 1998): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/Causality_and_Explanation_Wesley_Salmon_1997.pdf)

(Salmon, Probabilistic Causality, 1980): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/Probabilistic_Causality_Salmon_1980.pdf)

(Good, 1961): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/CausalCalculus_part_I_and_II_Good_1960.pdf)

(Mackie, 1980): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/cement-of-the-universe-a-study-of-causation-JL-Mackie-1980.pdf)

(Lewis, Counterfactuals, 1973): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/books/Counterfactuals-lewis-1973.pdf)

(Lewis, Causation, 1974): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/Lewis-Causation_1974.pdf)

(Susan Owicki, 1982): [here](https://github.com/dimitarpg13/root_cause_analysis_and_model_checking/blob/main/literature/ModelChecking/ProvingLivenessPropertiesOfConcurrentProgramsOwickiLamport1982.pdf)

1. In (Hansson & Jonsson, 1994) and (Susan Owicki, 1982) the symbol is used to denote the *leads-to* operator. In this document we use to denote *prima facie* cause. [↑](#footnote-ref-1)