Advanced Formal Tools 14x014
DTMC and PCTL

Stéphane Liem Nguyen
University of Geneva

stephane.nguyen@etu.unige.ch

February 23, 2023

This work by no means replaces proper lectures. We start by describing what a Kripke structure is, how to specify some properties we would like to verify using CTL. Later on, we describe what are Discrete Time Markov Chains and how to specify some properties using PCTL.

In general, we have this following scheme:

- Define model
- Define properties we would like to verify
- Verify properties (semantics)

1 CTL

1.1 Kripke Structure

A transition system is defined as: $\langle S, S^0, \rightarrow \rangle$. A Kripke Structure (model) is defined as: $\mathcal{K} = \langle S, S^0, \rightarrow, AP, L \rangle$ where:

- S is a finite set of states
- S^0 is a non empty set of initial states
- $\rightarrow \subseteq S \times S$ is a left total binary relation on S representing the transitions
- AP is a set of atomic propositions
- $L : S \rightarrow \mathcal{P}(AP)$ is a labeling function telling us what are the atomic propositions that are true in a given state

For example, S can contain all the possible states of a program (for some class of programs). We can then specify, using CTL formulas, what properties we would like to verify. For example, is it possible to reach a state we do not want ? (e.g writing at the same memory location at the same time)

1.2 Specifying properties using CTL formulas

Minimal syntax: Let's denote CTL to be the set of CTL formulas.

- $a \in AP \Rightarrow a \in CTL$
- $\Phi_1, \Phi_2 \in CTL \Rightarrow \neg\Phi_1, \Phi_1 \vee \Phi_2, \mathbf{EX}\Phi_1, \mathbf{EG}\Phi_1, \Phi_1\mathbf{EU}\Phi_2 \in CTL$

Extended syntax: Let's define $\Phi_1 \equiv \Phi_2 \iff \forall \mathcal{K}, [\![\Phi_1]\!]_\mathcal{K} = [\![\Phi_2]\!]_\mathcal{K}$ (see SMV course. Set of states satisfying Φ_1 are the same as the ones satisfying Φ_2 for any Kripke structure)

- $true \equiv a \vee \neg a$
- $false \equiv \neg true$
- $\Phi_1 \wedge \Phi_2 \equiv \neg(\neg\Phi_1 \vee \neg\Phi_2)$
- $\mathbf{AX}\Phi \equiv \neg\mathbf{EX}\neg\Phi$
- $\mathbf{AG}\Phi \equiv \neg\mathbf{EF}\neg\Phi$
- $\mathbf{EF}\Phi \equiv true\mathbf{EU}\Phi$
- $\mathbf{AF}\Phi \equiv \neg\mathbf{EG}\neg\Phi$
- $\Phi_1\mathbf{AU}\Phi_2 \equiv (\Phi_1\mathbf{AW}\Phi_2) \wedge \mathbf{AF}\Phi_2$ (all (paths/runs/executions) strong until. Φ_2 holds eventually.)
- $\Phi_1\mathbf{AW}\Phi_2 \equiv \neg(\neg\Phi_2\mathbf{EU}\neg(\Phi_1 \vee \Phi_2))$
- $\Phi_1\mathbf{EW}\Phi_2 \equiv (\Phi_1\mathbf{EU}\Phi_2) \vee (\mathbf{EG}\Phi_1)$ (exist (path/run) weak until)

Note that we can extend it even further (e.g $\implies$). The semantic is not extended by the extended syntax.

1.3 Denotational semantics

Of minimal syntax:

- $[\![a]\!] = \nu(a)$ if we have $\nu : AP \to \mathcal{P}(S)$ that gives the set of states satisfying the atomic proposition a. Otherwise, one can use the labeling function L to retrieve it.
- $[\![\neg\Phi_1]\!] = S \setminus [\![\Phi_1]\!]$
- $[\![\Phi_1 \vee \Phi_2]\!] = [\![\Phi_1]\!] \cup [\![\Phi_2]\!]$
- $[\![\mathbf{EX}\Phi]\!] = pre_\exists([\![\Phi]\!])$ (predecessors of states satisfying Φ)
- $[\![\mathbf{EG}\Phi]\!] = \nu Y.[\![\Phi]\!] \cap pre_\exists(Y)$ (greatest fixed point)
- $[\![\Phi_1\mathbf{EU}\Phi_2]\!] = \mu Y.[\![\Phi_2]\!] \cup ([\![\Phi_1]\!] \cap pre_\exists(Y))$

Remarks: To remember the formulas, $\mathbf{E}X, \mathbf{E}F, \mathbf{E}G$ use $pre_\exists$ (exist a path) while $\mathbf{A}X, \mathbf{A}F, \mathbf{A}G$ use $pre_\forall$ (see SMV course). Each time we see F (finally), we use an **union** (so least fixed point) and each time we see a G (globally), we use an **intersection** (so greatest fixed point) (because we want something globally true on the whole path).

For **AU** or **EU**, we have a least fixed point so we use μ instead of ν.

2 PCTL: Probabilistic real time CTL

CTL cannot verify properties that has :

- A time component (something that holds within some period of time)
- Probabilities (working on probabilistic models)

So PCTL is an extension of CTL but with a probability that something is true within some time units.

2.1 DTMC: Discrete Time Markov Chain

A DTMC can be defined as $\langle S, S^0, \mathcal{T} \rangle$.

Following **To add reference**, we define a **DTMC structure** to be $\langle S, S^0, \mathcal{T}, AP, L \rangle$ where:

- S is a finite set of states
- S^0 is a non empty set of initial states
- $\mathcal{T} : S \times S \to [0, 1]$ the transition probability function such that $\forall s, \sum_{s' \in S} T(s, s') = 1$.
- AP is a set of atomic propositions
- $L : S \to \mathcal{P}(AP)$ is a labeling function telling us what are the atomic propositions that are true in a given state

2.2 Specifying properties using PCTL formulas

We will write the formulas slightly differently than from the **REFERENCE TO ADD** so that it looks similar to what is used in PRISM but formulas were taken from the reference.

Minimal syntax: Let's denote $PCTL = \mathcal{SF} \cup \mathcal{PF}$ to be the set of PCTL formulas where $\mathcal{SF}$ is the set of state formulas (see **reference TO ADD**) and $\mathcal{PF}$ is the set of path formulas (see **reference TO ADD**).

- $a \in AP \Rightarrow a \in \mathcal{SF}$
- $\Phi_1, \Phi_2 \in \mathcal{SF} \Rightarrow \neg\Phi_1, \Phi_1 \vee \Phi_2, \Phi_1 \wedge \Phi_2, \Phi_1 \to \Phi_2 \in \mathcal{SF}$. Note that $\wedge$ and $\to$ are not necessary.
- $\Phi_1, \Phi_2 \in \mathcal{SF}$ and $t \in \mathbb{N} \cup \{\infty\} \Rightarrow (\Phi_1 \mathbf{U}^{\leq t} \Phi_2), (\Phi_1 \mathbf{W}^{\leq t} \Phi_2) \in \mathcal{PF}$[1] where **U** is for strong until and **W** for weak until
- $\Phi \in \mathcal{PF}$ and $p \in [0, 1] \Rightarrow P_{\geq p}[\Phi], P_{>p}[\Phi] \in \mathcal{SF}$ (or $P[\Phi] \geq p$ and $P[\Phi] > p$ respectively)

State formulas are for properties of states while path formulas are for properties of paths. Moreover, recall that we're working with Discrete Time Markov Chains so t represent t time units.

What about EX, EG and EU ? What about all the extended syntax of CTL ?

- **EX**$\Phi \equiv P_{>0}[\Phi \mathbf{U}^{\leq 2} true]$ **TO MODIFY, not in the reference, should also have > 0 in the U**
- **EG**$\Phi \equiv P_{>0}[\Phi \mathbf{W}^{\leq \infty} false]$ (proba at least 0 means that there exists a path)
- Φ_1**EU**$\Phi_2 \equiv P_{>0}[\Phi_1 \mathbf{U}^{\leq \infty} \Phi_2]$

But to define these above, we have to first define true and false. For the extended syntax (except "true", "false" and "and"), we can write the following equivalences (even though one can obtain them using the equivalences we defined in the CTL section)

- **AX**$\Phi \equiv P_{\geq 1}[\Phi \mathbf{U}^{\leq 2} true]$ **TO MODIFY, not in the reference, should also have > 0 in the U**
- **AG**$\Phi \equiv P_{\geq 1}[\Phi \mathbf{W}^{\leq \infty} false]$
- **EF**$\Phi \equiv P_{>0}[true \mathbf{U}^{\leq \infty} \Phi]$

[1] so called bounded variants of path properties. In PRISM, to get the unbounded version, remove $\leq t$. Otherwise, in this pdf, we can have $t = \infty$

- $\mathbf{AF}\Phi \equiv P_{\geq 1}[true\mathbf{U}^{\leq\infty}\Phi]$
- $\Phi_1\mathbf{AU}\Phi_2 \equiv P_{\geq 1}[\Phi_1\mathbf{U}^{\leq\infty}\Phi_2]$
- $\Phi_1\mathbf{AW}\Phi_2 \equiv P_{\geq 1}[\Phi_1\mathbf{W}^{\leq\infty}\Phi_2]$
- $\Phi_1\mathbf{EW}\Phi_2 \equiv P_{>0}[\Phi_1\mathbf{W}^{\leq\infty}\Phi_2]$

E (exists) and **A** (all) are hidden behind the probabilities.
So essentially, to go from **E** (exist) to **A** (all), we just need to change the probability from > 0 to ≥ 1.

So what can we do more than CTL ?

- Can have a "portion" of the paths instead of at least 1 or all paths (More freedom on the quantification over paths).
- Can specify properties that hold during a specify time interval (More freedom on the quantification over time).

To add some examples

Some questions we might ask: Can we also get a set of states that satisfy the properties ? Not just from the initial state ? Can we also verify properties starting from other states ? What are the probabilities ? (in PRISM, we use $P_{=?}$) What about expected cumulative reward ? (for Markov Reward Processes) What can we do in PRISM ?