CS 515: Temporal Logic and Model Checking

Stefan D. Bruda

Winter 2019

Kripke Structures

- Used to model systems with discrete states, just like computing systems
- Graph with vertices representing states (labelled with those elementary propositions that hold in the respective state) and unlabeled edges:
- A Kripke structure K over a set AP of atomic propositions is a tuple $(S, S_0, \rightarrow, L)$
  - S is a set of states
  - $S_0 \subseteq S$ is the set of initial states
  - $\rightarrow \subseteq S \times S$ is the transition relation
  - $L : S \rightarrow 2^{AP}$ is a function that assigns to each states exactly all the atomic propositions that are true in that state
- Usually assumed that $\rightarrow$ is total (for every state $s \in S$ there exists a state $t \in S$ such that $s \rightarrow t$)
- A path π in a Kripke structure K is a sequence $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ such that $s_i \rightarrow s_{i+1}$ for all $i \geq 0$
  - π^i refers to s_i, the i-th state on the path (with π^0 being the initial state)
- All the paths starting from a given state s_0 can be represented together as a computation tree with states as node labels
  - This tree is rooted at s_0 and (s, t) is an edge in the tree if and only if $s \rightarrow t$

Modelling Systems as Kripke Structures

- Modelling based on set of variables $V = \{v_1, v_2, \ldots, v_n\}$, each ranging over the domain D
  - Valuation for V: function that assigns a value from D to each variable in V
- A state of a system is described by a valuation $s : V \rightarrow D$
  - Given a valuation we can derive a conjunctive formula true for that valuation
  - The terms in the conjunction are the atomic propositions that are true in that state (though they are not necessarily the sole such atomic propositions)
  - Example: $V = \{v_1, v_2, v_e\}$ and $s(v_1) = 3$, $s(v_2) = 2$, $s(v_3) = 1$ imply the formula: $v_1 = 3 \wedge v_2 = 2 \wedge v_3 = 1$
  - Whenever the variable names are understood from the context (or irrelevant) we can give the valuation as an n-tuple
- A transition of the system is a switch from one valuation to another
  - We express the present state valuation over $V = \{v_1, \ldots, v_n\}$ and the next state valuation over the copy $V' = \{v'_1, \ldots, v'_n\}$ of V
- Major issue in modeling: granularity of transitions
  - Too coarse a granularity may skip states introducing errors
  - One must make sure that all transitions are atomic = no observable state can result from executing part of a transition
  - Too fine a granularity on the other hand may introduce states that are not reachable in the actual system but exist in the model and may introduce errors that in practice are not relevant

Temporal Logic

- Boolean (propositional) logic is used to reason about any given state (of a computation)
  - Boolean formulae consist of atomic propositions (from AP) put together using the operators $\wedge$, $\vee$, $\neg$
- This continues to be the case, but in addition we also need to reason about computation paths
- This is accomplished by using five temporal operators that establish some form of "before" and "after" relations:
  - X for a property that has to be true in the next state of the path
  - F for a property that has to eventually become true along the path
  - G for a property that has to hold in every state along the path
  - U for a property that has to hold continuously along a path until another property becomes true and remains true for the rest of the path
  - R for a property that has to hold along a path until another property becomes true and releases the first property from its obligation
- The path properties are put together using the universal quantifier A (for all paths) and the existential quantifier E (form some path)
- All of the above define the CTL* temporal logic

CTL* Syntax and Semantics

- Formally the syntax of CTL* is defined as follows:
  1. If $p \in$ AP then p is a state formula
  2. If f and g are state formulae then so are $\neg f$, $f \wedge g$, $f \vee g$
  3. If f is a path formula then E f and A f are state formulae
  4. If f is a state formula then f is also a path formula
  5. If f and g are part formulae then so are $\neg f$, $f \wedge g$, $f \vee g$, X f, F f, G f, f U g, f R g
- The semantics of CTL* is given by the satisfaction operator $\vDash$, as follows:
- $K, s \vDash f$ is true iff the state formula f is true in the state s of the Kripke structure K
  1. $K, s \vDash p$ for $p \in$ AP iff $p \in L(s)$
  2. $K, s \vDash \neg f$ iff $\neg(K, s \vDash f)$
  3. $K, s \vDash f \vee g$ iff $K, s \vDash f \vee K, s \vDash g$
  4. $K, s \vDash f \wedge g$ iff $K, s \vDash f \wedge K, s \vDash g$
  5. $K, s \vDash$ A f iff for every path π starting from s, $K, \pi \vDash f$
  6. $K, s \vDash$ E f iff there is a path π from s such that $K, \pi \vDash f$

CTL* Semantics (cont'd)

- $K, s \vDash \pi$ is true iff the path formula f is true along the path π in the Kripke structure K
  1. $K, \pi \vDash f$ iff $K, \pi^0 \vDash f$
  2. $K, \pi \vDash \neg f$ iff $\neg(K, \pi \vDash f)$
  3. $K, \pi \vDash f \vee g$ iff $K, \pi \vDash f \vee K, \pi \vDash g$
  4. $K, \pi \vDash f \wedge g$ iff $K, \pi \vDash f \wedge K, \pi \vDash g$
  5. $K, \pi \vDash$ X f iff $K, \pi^1 \vDash f$
  6. $K, \pi \vDash$ F f iff there exists $k \geq 0$ such that $K, \pi^k \vDash f$
  7. $K, \pi \vDash$ G f iff $K, \pi^k \vDash f$ for all $k \geq 0$
  8. $K, \pi \vDash f$ U g iff there exists $k \geq 0$ such that $K, \pi^k \vDash g$ and for all $0 \leq j \leq k$, $k, \pi^j \vDash f$
  9. $k, \pi \vDash f$ R g iff for all $j \geq 0$, if for every $i < j$ $k, \pi^i \nvDash f$ then $K, \pi^j \vDash g$
- Note in passing that $\{\vee, \neg, \text{X}, \text{U}, \text{E}\}$ is a complete set of operators for CTL*; indeed,

$$f \wedge g = \neg(\neg f \vee \neg g) \quad f \text{ R } g = \neg(\neg f \text{ U } \neg g) \quad \text{F } f = \mathit{true} \text{ U } f$$
$$\text{G } f = \neg\text{F } (\neg f) \qquad \text{A } f = \neg\text{E } (\neg f)$$

CTL and LTL

- Computation tree logic (CTL) is a restricted subset of CTL* in which each of the temporal operators must be immediately preceded by a quantifier
  - More formally, the restriction can be formulated by restricting the syntax of the path formulae to the following: If f and g are state formulae then X f, F f, G f, f U g, and f R g are path formulae
  - Ten basic temporal CTL operators: AX, EX, AF, EF, AG, EG, AU, EU, AR, ER
  - A complete set consists of only three operators (plus Boolean operators): EX, EG, and EU
- Linear-time temporal logic (LTL) is a restricted subset of CTL* where all the formulae have the form A f, where f is a path formula in which the only state formulae are atomic propositions
  - More formally, an LTL path formula is defined as follows:
    1. $p \in$ AP is a path formula
    2. If f and g are path formulae then so are $\neg f$, $f \vee g$, $f \wedge g$, X f, F f, G f, f U g, f R g
- CTL and LTL are not comparable
  - The LTL formula A(FG p) has no equivalent CTL formula
  - The CTL formula AG (EF p) has no equivalent LTL formula
- CTL* is strictly more expressive than both LTL and CTL
  - A (FG p) $\vee$ AG (EF p) is not expressible in either CTL or LTL

Complete Set of CTL Operators

- AX $f = \neg(\text{EX } \neg f)$
- EF $f =$ E true U f
- AG $f = \neg(\text{EF } \neg f)$
- AF $f = \neg(\text{EG } \neg f)$
- A f U $g = \neg(\text{E } \neg g \text{ U}(\neg f \wedge \neg g)) \wedge \neg(\text{EG } \neg g)$
- A f R $g = \neg(\text{E } \neg f \text{ U } \neg g)$
- E f R $g = \neg(\text{A } \neg f \text{ U } \neg g)$

Model Checking

- The model checking problem is formulated as follows:
  - Input: a Kripke structure $K = (S, S_0, \rightarrow, L)$, and a temporal logic formula f
  - Output: the set $S_f = \{s \in S : K, s \vDash f\}$
- The system K satisfies the specification f whenever $S_0 \subseteq S_f$ (all initial states satisfy f)
- The most intuitive model checking algorithm creates iteratively a set label(s) with all the subformulae of f that are true in the state s
  - Initially label(s) = $L(s)$ (atomic propositions)
  - At iteration i we process subformulae with $i-1$ nested CTL operators
  - Once done, we will have $K, s \vDash f$ iff $f \in$ label(s)
- This algorithm is suitable for CTL model checking
  - We consider inductively the five possible structures of a formula ($\neg f$, $f \vee g$, EX f, E f U g, and EG f).

CTL Model Checking

1. We add $\neg f$ to each set label(s) that does not contain f
2. We add $f \vee g$ to each set label(s) that contains either f or g
3. We add EX f to each set label(s) such that label(s') contains f, for some successor s' or s
   - The successors are found by following the outgoing edges of s
4. To add E f U g to the set of labels:
   1. Find all states labeled with g
   2. Follow $\rightarrow$ in reverse from each such a state looking for paths with all the states labeles with f; all these states are labeled with E f U g
5. The case EG f is based on the modified Kripke structure $K' = (S', S', \rightarrow', L)$, where $S' = \{s \in S : K, s \vDash f\}$, $\rightarrow' = \rightarrow \cap S' \times S'$
   - $K, s \vDash$ EG f iff $s \in S'$ and there exists a path in K' from s to some state t in a nontrivial strongly connected component of the graph $(S', \rightarrow')$
   - A strongly connected component (SCC) C is a maximal subgraph such that every node in C is reachable from every other node in C
   - C is nontrivial iff it either has more than one node or it contains one node and a self-loop

CTL Model Checking: E f U g

algorithm CheckEU(f, g):

1. $T \leftarrow \{s : g \in \text{label}(s)\}$
2. **for all** $s \in T$ **do** label(s) $\leftarrow$ label(s) $\cup \{\text{E } f \text{ U } g\}$
3. **while** $T \neq \{\}$ **do**
   1. **let** $s \in T$
   2. $T \leftarrow T \setminus \{s\}$
   3. **for all** t such that $t \rightarrow s$ **do**
      if E f U $g \notin$ label(t) **and** $f \in$ label(t) **then**
      1. label(t) $\leftarrow$ label(t) $\cup \{\text{E } f \text{ U } g\}$
      2. $T \leftarrow T \cup \{t\}$

- Complexity: $O(|S| + |\rightarrow|)$

CTL Model Checking: EG f

algorithm CheckFG(f):

1. $S' \leftarrow \{s : f \in \text{label}(s)\}$
2. $SCC \leftarrow \{C : C \text{ is a nontrivial SCC of } S'\}$
3. $T \leftarrow \bigcup_{C \in SCC} \{s : s \in C\}$
4. **for all** $s \in T$ **do** label(s) $\leftarrow$ label(s) $\cup \{\text{EG } f\}$
5. **while** $T \neq \{\}$ **do**
   1. **let** $s \in T$
   2. $T \leftarrow T \setminus \{s\}$
   3. **for all** t such that $t \in S'$ and $t \rightarrow s$ **do**
      if EG $f \notin$ label(t) **then**
      1. label(t) $\leftarrow$ label(t) $\cup \{\text{EG } f\}$
      2. $T \leftarrow T \cup \{t\}$

- Complexity: $O(|S| + |\rightarrow|)$
  - SCC can be found in $O(|S'| + |\rightarrow'|)$

Complexity of CTL Model Checking

- The algorithm consists of successively applying the state-labeling algorithms shown earlier starting from the most deeply nested sub-formulae of f and working our way outward inductively
- Each labeling phase (for each sub-formula) takes $O(|S| + | \rightarrow |)$
- Given that we have no more than $|f|$ operators in f it follows that the overall running time is $O(|f| \times (|S| + | \rightarrow |))$
  - That is linear with respect to the size of f and quadratic with respect to the number of states in K