NSWI101: SYSTEM BEHAVIOUR MODELS AND VERIFICATION

4. COMPUTATIONAL TREE LOGIC

Jan Kofroň
Computational Tree Logic (CTL)
CTL model checking
Comparison of CTL and LTL
Model Checking

System model

AG \(\text{start} \rightarrow \text{AF heat}\)

Property specification

Model Checker

Property satisfied

Property violated
MODEL CHECKING

System model

CTL

AG (start \rightarrow AF \text{heat})

Property specification

Model Checking

Property satisfied

Property violated
Another temporal logic, differing from LTL in expressive power

*Computational tree* refers to ability to properties of computational subtrees
(branching)

- as opposed to LTL that considers particular paths in isolation

The semantic model similar to LTL – also defined upon infinite paths of Kripke structure
Let AP be set of atomic propositions (Boolean variables).

CTL formulae are finite expressions created by following rules:
- $\top, \bot, p \in AP$ are CTL formulae

If $\varphi, \psi$ are CTL formulae, then the following are also CTL formulae:
- $AX \varphi$
- $AF \varphi$
- $AG \varphi$
- $A[\varphi \ U \psi]$
- $EX \varphi$
- $EF \varphi$
- $EG \varphi$
- $E[\varphi \ U \psi]$

Operators X, F, G, U have similar meaning as in LTL
Quantifiers A, E refer to paths – “all paths” vs. “exists path”
Let $M = (S, R, L)$ be Kripke structure

- $\langle s \rightarrow t \rangle$ denotes transition from state $s$ to state $t$
- $\langle s_0 \rightarrow \rightarrow \rangle$ denotes infinite path $\langle s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rangle$ starting at state $s_0$
CTL SEMANTICS

\[(M, s \models T) \land (M, s \not\models \bot)\]
\[(M, s \models p) \iff (p \in L(s))\]
\[(M, s \models \neg \varphi) \iff (M, s \not\models \varphi)\]
\[(M, s \models \varphi_1 \lor \varphi_2) \iff ((M, s \models \varphi_1) \lor (M, s \models \varphi_2))\]
\[(M, s \models AX \varphi) \iff (\forall \langle s \to t \rangle (M, t \models \varphi))\]
\[(M, s \models EX \varphi) \iff (\exists \langle s \to t \rangle (M, t \models \varphi))\]
\[(M, s \models AG \varphi) \iff (\forall \langle s \to \rangle \forall i \geq 0 : M, s_i \models \varphi)\]
\[(M, s \models EG \varphi) \iff (\exists \langle s \to \rangle \forall i \geq 0 : M, s_i \models \varphi)\]
\[(M, s \models AF \varphi) \iff (\forall \langle s \to \rangle \exists i \geq 0 : M, s_i \models \varphi)\]
\[(M, s \models EF \varphi) \iff (\exists \langle s \to \rangle \exists i \geq 0 : M, s_i \models \varphi)\]
\[(M, s \models A[\varphi_1 U \varphi_2]) \iff (\forall \langle s \to \rangle \exists i \geq 0 : (M, s_i \models \varphi_2) \land (\forall (j < i)(M, s_j \models \varphi_1)))\]
\[(M, s \models E[\varphi_1 U \varphi_2]) \iff (\exists \langle s \to \rangle \exists i \geq 0 : (M, s_i \models \varphi_2) \land (\forall (j < i)(M, s_j \models \varphi_1)))\]
CTL MODEL CHECKING

Based on identifying states of model satisfying sub-formulae of property formula:

1. Create derivation tree of property formula.
2. In bottom-up manner identify all states of model satisfying sub-formula associated with each node of derivation tree.
CTL MODEL CHECKING

Based on identifying states of model satisfying sub-formulae of property formula:

1. Create derivation tree of property formula.
2. In bottom-up manner identify all states of model satisfying sub-formula associated with each node of derivation tree.

\[
EG (E[p \cup q]) \land \neg (EX r)
\]

\[
EG (E[p \cup q])
\]

\[
E[p \cup q]
\]

\[
p
\]

\[
q
\]

\[
\neg (EX r)
\]

\[
EX r
\]

\[
r
\]
CTL formula can be transformed to contain just $\neg$, $\land$, EG, EX, and EU operators

Various algorithms for identification of states satisfying particular sub-formulae exist

- **explicit model checking** – explicit representation of each state in memory
- **symbolic model checking** – representing sets of states by Boolean formulae
Identification of states satisfying particular sub-formulae:

- operators $\neg$, $\land$, and $\text{EX}$ are trivial
- operators $\text{EG}$ and $\text{EU}$ require more complex algorithms
function CHECKEU(\(\varphi_1, \varphi_2\))

\[T := \{s : \varphi_2 \in \text{label}(s)\}\]

\[\text{for all } s \in T \text{ do}\]

\[\text{label}(s) := \text{label}(s) \cup \{E[\varphi_1 U \varphi_2]\}\]

\[\text{end for}\]

\[\text{while } T \neq \{\} \text{ do}\]

\[\text{choose } s \in T; T := T \setminus \{s\}\]

\[\text{for all } t : R(t, s) \text{ do}\]

\[\text{if } E[\varphi_1 U \varphi_2] \notin \text{label}(t) \land \varphi_1 \in \text{label}(t) \text{ then}\]

\[\text{label}(t) := \text{label}(t) \cup \{E[\varphi_1 U \varphi_2]\}\]

\[T := T \cup \{t\}\]

\[\text{end if}\]

\[\text{end for}\]

\[\text{end while}\]

\[\text{end function}\]
function CHECKEG(\(\varphi_1\))

\(S' := \{s : \varphi_1 \in label(s)\}\)

\(SCC = \{C : C\) is non-trivial SCC of \(S'\}\)

\(T := \bigcup_{C \in SCC} \{s : s \in C\}\)

for all \(s \in T\) do

\(\text{label}(s) := \text{label}(s) \cup \{\text{EG } \varphi_1\}\)

end for

while \(T \neq \{\}\) do

choose \(s \in T; T := T \setminus \{s\}\)

for all \(t : t \in S' \land R(t, s)\) do

if \(\text{EG } \varphi_1 \notin \text{label}(t)\) then

\(\text{label}(t) := \text{label}(t) \cup \{\text{EG } \varphi_1\}\)

\(T := T \cup \{t\}\)

end if

end for

end while

end function
**Explicit CTL Model Checking: Complexity**

Computing states satisfying particular sub-formulae:

- **CheckEU**: $O(|S| + |R|)$
- **CheckEG**: $O(|S| + |R|)$
  - Finding strongly connected components using Tarjan algorithm: $O(|S'| + |R'|)$
- **EX**: $O(|S| + |R|)$
- Negation and conjunction: $O(|S|)$
- $\varphi$ contains at most $|\varphi|$ different sub-formulae

**Total time complexity**: $O(|\varphi| \times (|S| + |R|))$
**DIFFERENCE BETWEEN CTL AND LTL**

CTL and LTL are incomparable

- there are properties of one logic not expressible in the other one
- difference stems from their different semantics – while CTL captures sub-trees of computational tree, LTL considers each path in isolation
- both are useful, each in different settings
Theorem: There is no LTL formula equivalent to CTL formula $\text{AG (EF } p\text{)}$. 
Theorem: There is no LTL formula equivalent to CTL formula $AG(\text{EF } p)$.

Proof:

1. For contradiction assume there exists LTL formula $\varphi$ equivalent to $AG(\text{EF } p)$.
2. State $s_0$ of KS(1) satisfies $AG(\text{EF } p)$. Therefore, $s_0$ satisfies $\varphi$.
3. Since $\varphi$ is satisfied in $s_0$, path looping in $s_0$ also satisfies it.
4. Therefore, state $s_0$ of KS(2) also satisfies $\varphi$.
5. Since $AG(\text{EF } p)$ and $\varphi$ are equivalent, state $s_0$ of KS(2) also satisfies $AG(\text{EF } p)$, which is contradiction.
**Theorem:** There is no CTL formula equivalent to LTL formula $F (G p)$. 
- in particular it is not equivalent to $AF (AG p)$
Theorem: There is no CTL formula equivalent to LTL formula $F(G\, p)$. In particular it is not equivalent to $AF(AG\, p)$.

Proof:
- Consider Kripke structure below whose state $s_0$ satisfies $F(G\, p)$, but does not satisfy $AF(AG\, p)$.
\[ M, s_0 \models F(Gp) \]

\[ M, s_0 \not\models AF(AGp) \]
Consider producer and consumer communicating over reliable network:

Is this CTL formula satisfied in "OK" state?

$$\text{AG} (\text{Sent} = \Rightarrow \text{AF} (\text{OK}))$$

No! Paths infinitely looping in "waiting" states violate it.
Consider producer and consumer communicating over reliable network:

Is this CTL formula satisfied in “OK” state?

\[ \text{AG} (\text{Sent} \implies \text{AF} (\text{OK})) \]
Consider producer and consumer communicating over reliable network:

Is this CTL formula satisfied in “OK” state?

\[ \text{AG} \left( \text{Sent} \implies \text{AF} \left( \text{OK} \right) \right) \]

\textbf{No!} Paths infinitely looping in “waiting” states violate it.
We can define *fairness constraint* to avoid this types of failures:
We can define *fairness constraint* to avoid this types of failures:

- Introducing new AP *Idle*
- Specifying *fairness constraint* as \( \neg \text{Idle} \)
- When model checking, only *fair paths* considered – those containing infinitely many fair states.
Fair paths: introducing new atomic proposition \textit{fair}:

- \textit{fair} is true in state \( s \) if there exists fair path starting in \( s \)
- \( M, s \models_F EG \) true
- Determining fair paths and deciding upon \( EG \ p \) formulae

\begin{align*}
M, s \models_F p & \iff M, s \models p \land \text{fair} \\
M, s \models_F EX \varphi & \iff M, s \models EX (\varphi \land \text{fair}) \\
M, s \models_F E[\varphi \ U \psi] & \iff M, s \models E[\varphi \ U (\psi \land \text{fair})]
\end{align*}
Complexity of CTL and LTL explicit model checking differs a bit:

- CTL: $O(|M| \times |\varphi|)$
- LTL: $O(|M| \times 2^{|\varphi|})$

Both linear in size of model, LTL exponential in size of formula

- Practically negligible difference as formula is usually much smaller than model