NSWI101: System Behaviour Models and Verification

4. Computational Tree Logic

Jan Kofroň

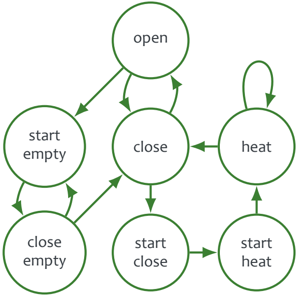
FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Department of
Distributed and
Dependable
Systems

Today

- Computational Tree Logic (CTL)
- CTL model checking
- Comparison of CTL and LTL

MODEL CHECKING



System model

AG (start $\rightarrow$ AF heat)

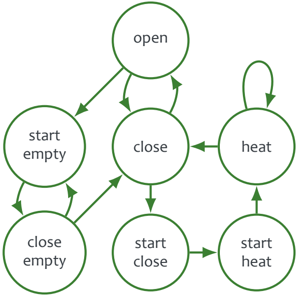
Property specification

Model Checker

Property satisfied

Property violated

MODEL CHECKING



System model

CTL

AG (start → AF heat)

Property specification

Model Checking

Model Checker

Property satisfied

Property violated

COMPUTATIONAL TREE LOGIC – CTL

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

CTL syntax

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 φ, ψ are CTL formulae, then the following are also CTL formulae:

- AX φ
- AF φ
- AG φ
- A[φ U ψ]
- EX φ
- EF φ
- EG φ
- E[φ U ψ]

Operators X, F, G, U have similar meaning as in LTL
Quantifiers A, E refer to paths – “all paths” vs. “exists a path”

CTL SEMANTICS

Let $M = (S, R, L)$ be Kripke structure

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

CTL SEMANTICS

$$(M, s \models \top) \wedge (M, s \not\models \bot)$$

$$(M, s \models p) \Leftrightarrow (p \in L(s))$$

$$(M, s \models \neg\varphi) \Leftrightarrow (M, s \not\models \varphi)$$

$$(M, s \models \varphi_1 \vee \varphi_2) \Leftrightarrow ((M, s \models \varphi_1) \vee (M, s \models \varphi_2))$$

$$(M, s \models \mathrm{AX}\,\varphi) \Leftrightarrow (\forall\langle s \to t\rangle(M, t \models \varphi))$$

$$(M, s \models \mathrm{EX}\,\varphi) \Leftrightarrow (\exists\langle s \to t\rangle(M, t \models \varphi))$$

$$(M, s \models \mathrm{AG}\,\varphi) \Leftrightarrow (\forall\langle s_0 \longrightarrow\rangle\forall i \geq 0 : M, s_i \models \varphi)$$

$$(M, s \models \mathrm{EG}\,\varphi) \Leftrightarrow (\exists\langle s_0 \longrightarrow\rangle\forall i \geq 0 : M, s_i \models \varphi)$$

$$(M, s \models \mathrm{AF}\,\varphi) \Leftrightarrow (\forall\langle s_0 \longrightarrow\rangle\exists i \geq 0 : M, s_i \models \varphi)$$

$$(M, s \models \mathrm{EF}\,\varphi) \Leftrightarrow (\exists\langle s_0 \longrightarrow\rangle\exists i \geq 0 : M, s_i \models \varphi)$$

$$(M, s \models \mathrm{A}[\varphi_1\,\mathrm{U}\,\varphi_2]) \Leftrightarrow (\forall\langle s_0 \longrightarrow\rangle\exists i \geq 0 : (M, s_i \models \varphi_2) \wedge (\forall(j < i)(M, s_j) \models \varphi_1))$$

$$(M, s \models \mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2]) \Leftrightarrow (\exists\langle s_0 \longrightarrow\rangle\exists i \geq 0 : (M, s_i \models \varphi_2) \wedge (\forall(j < i)(M, s_j) \models \varphi_1))$$

CTL MODEL CHECKING

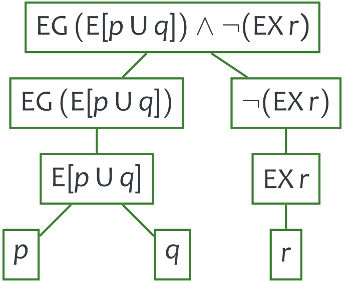
Based on identifying states of model satisfying sub-formulae of property formula:

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.



CTL MODEL CHECKING

CTL formula can be transformed to contain just $\neg$, $\wedge$, 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

Explicit CTL model checking

Identification of states satisfying particular sub-formulae:

- operators $\neg$, $\wedge$, and EX are trivial
- operators EG and EU require more complex algorithms

Explicit CTL model checking: EU operator

```
function CheckEU(φ1, φ2)
    T := {s : φ2 ∈ label(s)}
    for all s ∈ T do
        label(s) := label(s) ∪ {E[φ1 U φ2]}
    end for
    while T ≠ {} do
        choose s ∈ T; T := T \ {s}
        for all t : R(t, s) do
            if E[φ1 U φ2] ∉ label(t) ∧ φ1 ∈ label(t) then
                label(t) := label(t) ∪ {E[φ1 U φ2]}
                T := T ∪ {t}
            end if
        end for
    end while
end function
```

Explicit CTL model checking: EG operator

```
function CheckEG(φ₁)
    S' := {s : φ₁ ∈ label(s)}
    SCC = {C : C is non-trivial SCC of S'}
    T := ⋃_{C∈SCC} {s : s ∈ C}
    for all s ∈ T do
        label(s) := label(s) ∪ {EG φ₁}
    end for
    while T ≠ {} do
        choose s ∈ T; T := T \ {s}
        for all t : t ∈ S' ∧ R(t, s) do
            if EG φ₁ ∉ label(t) then
                label(t) := label(t) ∪ {EG φ₁}
                T := T ∪ {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: $O(|S|)$
- φ contains at most $|\varphi|$ different sub-formulae

Total time complexity: $O(|\varphi| * (|S| + |R|))$

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

CTL $\not\subseteq$ LTL

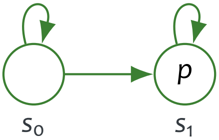
Theorem: There is no LTL formula equivalent to CTL formula AG (EF p).

CTL $\not\subseteq$ LTL

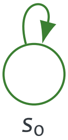
Theorem: There is no LTL formula equivalent to CTL formula AG (EF p).

Proof:

1. For contradiction assume there exists LTL formula φ equivalent to AG (EF p).
2. State s_0 of KS(1) satisfies AG (EF p). Therefore, s_0 satisfies φ.
3. Since φ is satisfied in s_0, path looping in s_0 also satisfies it.
4. Therefore, state s_0 of KS(2) also satisfies φ.
5. Since AG (EF p) and φ are equivalent, state s_0 of KS(2) also satisfies AG (EF p), which is contradiction.



KS(1)



KS(2)

LTL $\not\subseteq$ CTL

Theorem: There is no CTL formula equivalent to LTL formula F (G *p*).

- in particular it is not equivalent to AF (AG *p*)

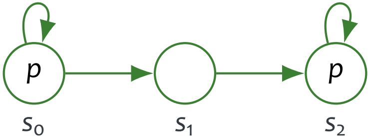
LTL $\not\subseteq$ CTL

Theorem: There is no CTL formula equivalent to LTL formula F (G p).

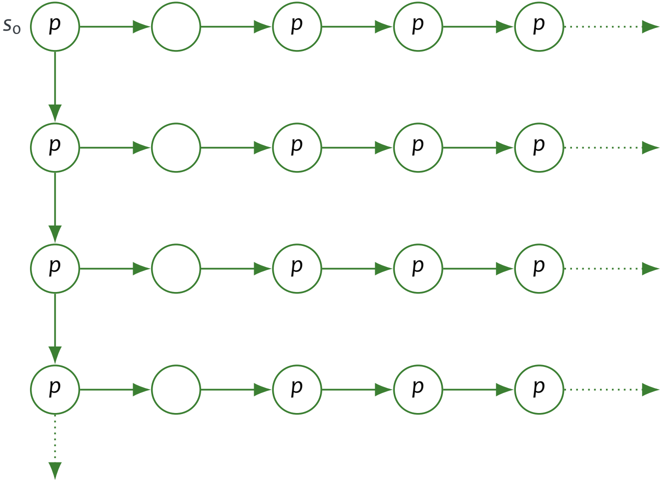
- in particular it is not equivalent to AF (AG p)

Proof:

- Consider Kripke structure below whose state s_0 satisfies F (G p), but does not satisfy AF (AG p).



Computational-tree perspective

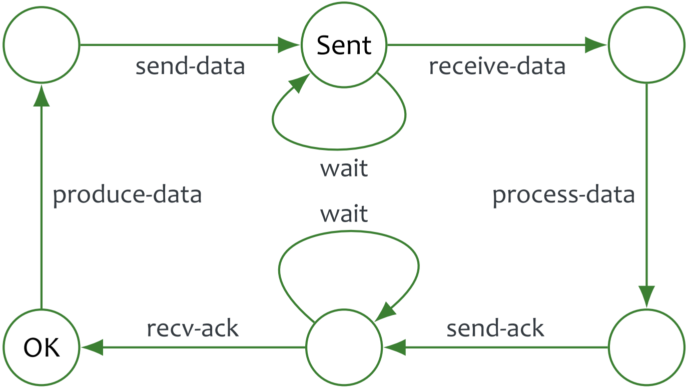


$$M, s_0 \models \mathsf{F}\,(\mathsf{G}\,p)$$

$$M, s_0 \not\models \mathsf{AF}\,(\mathsf{AG}\,p)$$

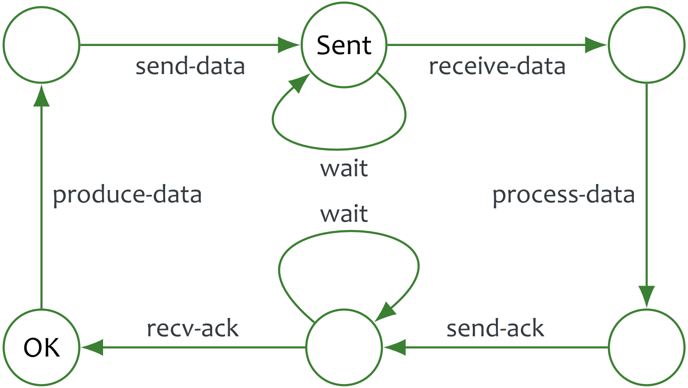
FAIRNESS CONSTRAINTS

Consider producer and consumer communicating over reliable network:



FAIRNESS CONSTRAINTS

Consider producer and consumer communicating over reliable network:

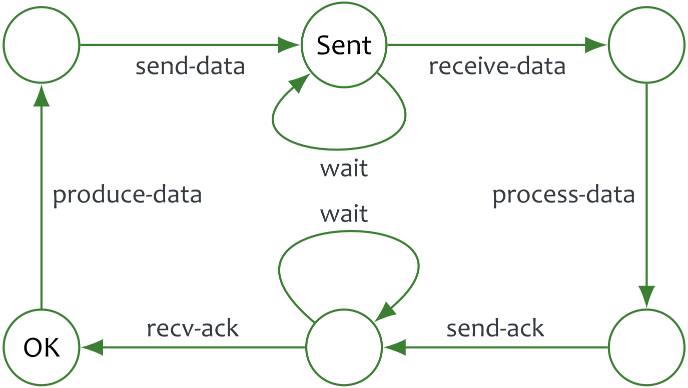


Is this CTL formula satisfied in "OK" state?

$$\mathrm{AG}\,(Sent \implies \mathrm{AF}\,(OK))$$

FAIRNESS CONSTRAINTS

Consider producer and consumer communicating over reliable network:



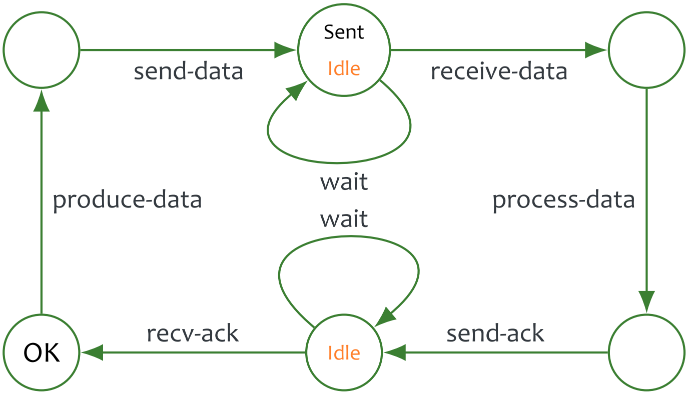
Is this CTL formula satisfied in "OK" state?

$$\mathrm{AG}\,(\mathit{Sent} \implies \mathrm{AF}\,(\mathit{OK}))$$

No! Paths infinitely looping in "waiting" states violate it.

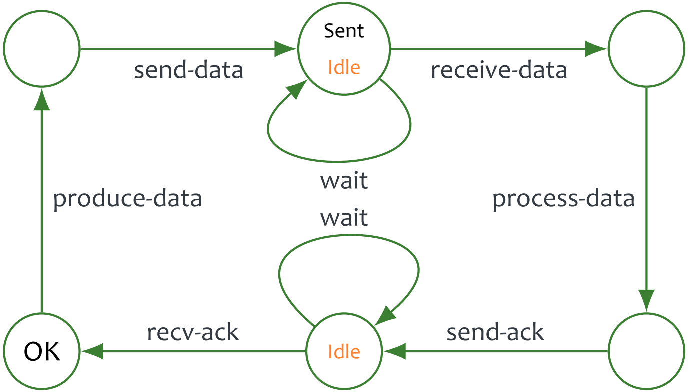
Fairness constraints

We can define *fairness constraint* to avoid this types of failures:



FAIRNESS CONSTRAINTS

We can define *fairness constraint* to avoid this types of failures:



Introducing new AP Idle

Specifying *fairness contraint* as ¬*Idle*

When model checking, only *fair paths* considered – those containing infinitely many fair states.

Fairness constraints

Fair paths: introducing new atomic proposition *fair*:

- *fair* is true in state $s \leftrightarrow$ there exists fair path starting in s
- $M, s \models_F \mathrm{EG}\, true$
- Determining fair paths and deciding upon EG p formulae

- $M, s \models_F p \leftrightarrow M, s \models p \wedge fair$
- $M, s \models_F \mathrm{EX}\, \varphi \leftrightarrow M, s \models \mathrm{EX}\, (\varphi \wedge fair)$
- $M, s \models_F \mathrm{E}[\varphi\, \mathrm{U}\, \psi] \leftrightarrow M, s \models \mathrm{E}[\varphi\, \mathrm{U}\, (\psi \wedge fair)]$

COMPLEXITY OF ALGORITHMS

Complexity of CTL and LTL explicit model checking differs a bit:

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

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

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