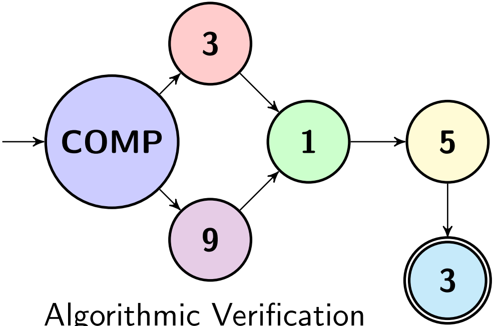


Algorithmic Verification

Expressiveness, CTL Model Checking

Dr. Liam O'Connor
CSE, UNSW (for now)
Term 1 2020

Comparing Logics

Formula Equivalence

Two formulae are equivalent iff they admit the same models.

$$\frac{\forall A.\ (A \models P) \Leftrightarrow (A \models Q)}{P \equiv Q}$$

Logic Expressiveness

A logic L_1 is *more expressive* than a logic L_2, written $L_2 \subseteq L_1$, iff:

For all $\varphi_2 \in L_2$, *there is a* $\varphi_1 \in L_1$ *such that* $\varphi_1 \equiv \varphi_2$.

CTL $\subseteq$ CTL*? LTL $\subseteq$ CTL*? *LTL* $\subseteq$ CTL? *CTL* $\subseteq$ LTL?

LTL $\subseteq$ CTL*

LTL formulae look like CTL* *path formulae*. How do we convert them into equivalent *state formulae*?

Recall that $A \models \varphi$ iff $\forall \rho \in \mathsf{Traces}(A).\ \rho \models \varphi$

For all LTL formulae φ:

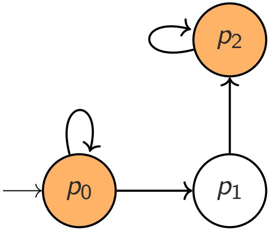
$$A \models_{\mathsf{LTL}} \varphi \iff A \models_{\mathsf{CTL}^*} \mathbf{A}\ \varphi$$

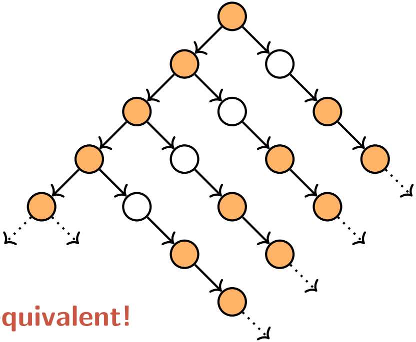
Proof follows trivially from the definition of **A**.

CTL ⊆ LTL?

CTL Formula: **AF AG** ●

LTL Formula: **FG** ●? does this work?





It's not equivalent!

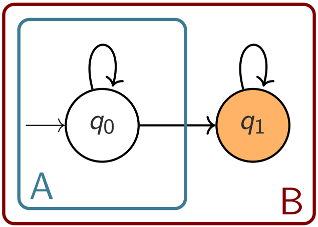
CTL $\not\subseteq$ LTL

Let's prove it.

Lemma (Trace Inclusion)

If $\text{Traces}(A) \subseteq \text{Traces}(B)$ then for any LTL formula φ,
$B \models \varphi \implies A \models \varphi$

Suppose $\exists$ an LTL formula φ that is equivalent to **AG EF** ●.



Proof

Observe that $B \models$ **AG EF** ● but $A \not\models$ **AG EF** ●

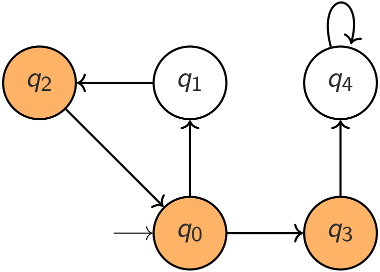
Because φ is equivalent, we know $B \models \varphi$ and $A \not\models \varphi$.

But, as $\text{Traces}(A) \subseteq \text{Traces}(B)$, our lemma says that $A \models \varphi$.

Contradiction!

LTL $\subseteq$ CTL?

LTL Formula: **F** (● $\wedge$ **X** ●)
CTL Formula: **AF** (● $\wedge$ **AX** ●). Does this work?



Nope!

LTL $\not\subseteq$ CTL

Lemma

It is possible to construct two families of automata A_i and B_i such that:

- They are distinguished by the LTL formula $\mathbf{F}\ \mathbf{G} \bullet$, that is: $A_i \models \mathbf{F}\ \mathbf{G} \bullet$ but $B_i \not\models \mathbf{F}\ \mathbf{G} \bullet$ for any i.
- They cannot be distinguished by CTL formulae of length $\leq i$. That is, $\forall i.\ \forall \varphi.\ |\varphi| \leq i \Rightarrow (A_i \models \varphi \Leftrightarrow B_i \models \varphi)$

See the textbook (Baier and Katoen) for details.

Proof

Let φ be a CTL formula equivalent to $\mathbf{F}\ \mathbf{G} \bullet$.Let k be the length of φ, i.e. $k = |\varphi|$. From lemma, $A_k \models \mathbf{F}\ \mathbf{G} \bullet$ and $B_k \not\models \mathbf{F}\ \mathbf{G} \bullet$, but also $A_k \models \varphi \Leftrightarrow B_k \models \varphi$, so φ cannot be equivalent.
Contradiction!

CTL $\subset$ CTL*

Every CTL formula is also a CTL* formula. But is it a strict inclusion (i.e. CTL $\subset$ CTL*)?

Yes. We know already that LTL $\subseteq$ CTL* and that LTL $\not\subseteq$ CTL. So pick any LTL formula that cannot be expressed in CTL, and we have a formula that cannot be expressed in CTL but can be in CTL*.

LTL $\subset$ CTL*

We saw that LTL $\subseteq$ CTL*. But is it a strict inclusion? (i.e. LTL $\subset$ CTL*)?

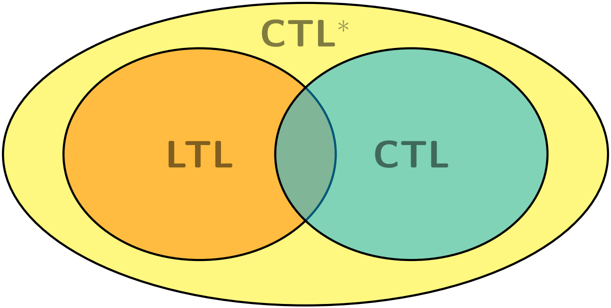
Yes. We know already that CTL $\subseteq$ CTL* and that CTL $\not\subseteq$ LTL. So pick any CTL formula that cannot be expressed in LTL, and we have a formula that cannot be expressed in LTL but can be in CTL*.

(LTL ∪ CTL) ⊂ CTL*

Is there any formula that **can** be expressed in CTL* but not in CTL nor in LTL?

Strict Inclusion

Yes. The proof is very involved, but the formula **E G F** ● cannot be expressed in either LTL nor CTL.



The CTL Model Checking Problem

Given

- A CTL formula φ, and
- An automaton A,

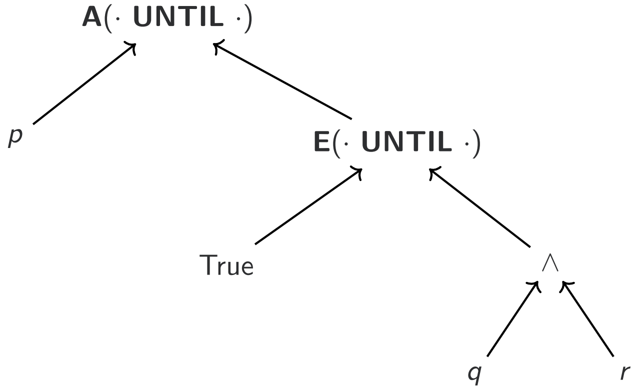
Determine if $A \models \varphi$.

Our approach

We first break the formula up into a *parse tree*. Then, annotate states in a bottom-up fashion with the (sub-)formulae they satisfy.

Parse Trees

$$\mathbf{A}(p \ \mathbf{UNTIL}\ \mathbf{E}(\text{True}\ \mathbf{UNTIL}\ q \wedge r))$$



Formal Algorithm: Basic Propositions

```
case φ ∈ 𝒫 do                                /* Atomic proposition */
    foreach q ∈ Q do
        if φ ∈ L(q) then
            q.φ := True;
        else
            q.φ := False;

case φ = ¬ψ do                                /* Negation */
    Mark(A, ψ) ;
    foreach q ∈ Q do
        q.φ := ¬q.ψ ;

case φ = ψ1 ∧ ψ2 do                           /* Conjunction */
    Mark(A, ψ1); Mark(A, ψ2) ;
    foreach q ∈ Q do
        q.φ := q.ψ1 ∧ q.ψ2 ;
```

Formal Algorithm: EX

```
case φ = EX ψ do                          /* Exists a Successor */
    Mark(A, ψ) ;
    foreach q ∈ Q do
        q.φ := False;
    foreach (q, q′) ∈ δ do
        if q′.ψ then
            q.φ := True ;
```

We can simplify **AX** ψ to $\neg$**EX** $\neg\psi$. Why?

```
case φ = E ψ1 UNTIL ψ2 do                    /* Exist Until */
    Mark(A, ψ1) ; Mark(A, ψ2) ;
    foreach q ∈ Q do
        q.φ := False;
        q.visited := False;
        if q.ψ2 then
            q.φ := True ;
            q.visited := True ;
            W := W ∪ {q};
    while W ≠ ∅ do
        q := pop(W);  /* q satisfies φ */
        foreach (q′, q) ∈ δ do
            if ¬q′.visited then
                q′.visited := True ;
                if q′.ψ1 then
                    q′.φ := True;  W := W ∪ {q′};
```

```
case φ = A ψ1 UNTIL ψ2 do                    /* For All Until */
    Mark(A, ψ1) ; Mark(A, ψ2);
    foreach q ∈ Q do
        q.φ := False;
        q.nbUnchecked := |δ(q)|;
        if q.ψ2 then
            q.φ := True ;
            W := W ∪ {q};
    while W ≠ ∅ do
        q := pop(W);
        /* q satisfies φ */
        foreach (q′, q) ∈ δ do
            q′.nbUnchecked := q′.nbUnchecked − 1 ;
            if (q′.nbUnchecked = 0 ∧ q′.ψ1 ∧ ¬q′.φ) then
                q′.φ := True ;
                W := W ∪ {q′};
```

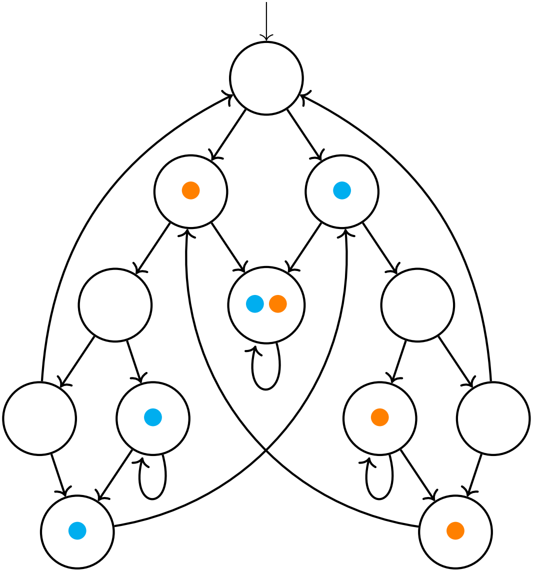
Complexity?

Assume a fixed size of formula $|\varphi|$, what is the run time complexity of this algorithm?

- Complexity for atomic propositions, $\wedge$ and $\neg$: $\mathcal{O}(|Q|)$
- Complexity for **EX**: $\mathcal{O}(|Q|)$
- Complexity for **E**($\cdot$ **UNTIL** $\cdot$): $\mathcal{O}(|Q| + |\delta|)$
- Complexity for **A**($\cdot$ **UNTIL** $\cdot$): $\mathcal{O}(|Q| + |\delta|)$

Therefore, overall complexity is: $\mathcal{O}(\ (|Q| + |\delta|) \times |\varphi|\ )$

Example



Procedure

- **Simplify** to basic CTL operations.
- **Build** parse tree for new formula.
- **Mark** states bottom up as described.

Example

- **EF** (• ∧ •)
- **EF AG** (• ∧ •)

Bibliography

Expressiveness:

- Huth/Ryan: Logic in Computer Science, Section 3.5
- Baier/Katoen: Principles of Model Checking, Section 6.3

CTL Model Checking

- Bérard et al: System and Software Verification, Section 3.1
- Baier/Katoen: Principles of Model Checking, Section 6.4
- Clarke et al: Model Checking, Section 4.1
- Huth/Ryan: Logic in Computer Science, Section 3.6