Overview

- **Specification**
  - Linear Time Logic
    - Describes behavior along infinite paths
  - Computation Tree Logic
    - Describes behavior that is possible starting from a state
  - CTL*
    - Relaxes CTL, encompasses both LTL and CTL
    - Strictly more expressive than CTL and LTL

Model Checking

- **Modelling**
  - Deterministic finite automata
    - In each state exactly one outgoing transition for every possible label
  - Nondeterministic finite automata
    - Any finite number of outgoing transitions for each state and label permitted
  - Büchi automata
    - Accepting condition on infinite runs
  - Kripke structures
    - Set of labels on the states rather than on transitions. No final states, no acceptance condition.
CTL model checking

- A straightforward labelling algorithm for CTL
- Given a Kripke structure $M = (S, s_0, \rightarrow, \mu)$

Mutual exclusion example:

Safety: None of the two processes should be in the critical section at the same time.

Fairness: Each process should be able to enter the critical section.

Atomic propositions:

- $p_{c1} = 0$
- $p_{c1} = 1$
- $p_{c1} = 2$
- $p_{c1} = 3$
- $0_1$
CTL model checking

A labelling algorithm of CTL

Given a Kripke structure $M=(S, s_0, \rightarrow, \mu)$

Given a CTL specification

Convert it to ENF

Syntax

$\phi ::= p \mid \neg \phi \mid \phi_1 \lor \phi_2 \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A(\phi_1 U \phi_2) \mid E(\phi_1 U \phi_2)$

Every CTL formula can be translated into Existential Normal Form (ENF)

$\phi ::= p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid EX\phi \mid E(\phi_1 U \phi_2) \mid EG\phi$
Example

\[
\text{AG AF } \neg(\mu_1 = 1 \land \mu_2 = 1)
\]

is equivalent to

\[
\neg E[ \text{true} U \text{EG} (\mu_1 = 1 \land \mu_2 = 1)]
\]

For all sub-formulas label states that satisfy them

- Recursive bottom-up computation:
  - consider the parse-tree of \( \phi \)
  - start with atomic propositions \( p \) in the leaves of the tree
  - for all states \( s \) if \( p \in \mu(s) \) add \( p \) to the labels of \( s \)
  - go one level up in the tree and check sub-formula
    - if subformula is true in \( s \), add it to \( label(s) \)
    - proceed until the root of the tree is checked

- \( M \models \phi \) if the initial state is labelled \( \phi \)

Example

\[
\neg E[ \text{true} U \text{EG} (\mu_1 = 1 \land \mu_2 = 1)]
\]

Consider the parse tree
Let $\text{label}(s)$ the set of labels of state $s$
- Initially $\text{label}(s) = \{\text{true}\}$
- Given a sub-formula $\phi$ in ENF there are six cases to consider

$$\phi ::= p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid \text{EX}\phi \mid \text{E}(\phi_1 U \phi_2) \mid \text{EG}\phi$$

**Labelling algorithm**

**Case 1:** $\phi \in \text{AP}$
- Add $\phi$ to labels of $s$ if $\phi \in \mu(s)$

**Case 2:** $\phi$ is of the form $\neg \psi$
- Add $\phi$ to labels of $s$ if $\psi \notin \text{labels}(s)$

**Case 3:** $\phi$ is of the form $\phi_1 \land \phi_2$
- Add $\phi$ to labels of $s$ if $\phi_1, \phi_2 \in \text{labels}(s)$
Labelling algorithm

Case 4: $\phi$ is of the form $\text{EX} \varphi$

Add $\phi$ to labels of $s$ if

$$\exists (s, s') \in \rightarrow \text{ such that } \varphi \in \text{labels}(s')$$

Labelling algorithm

Case 5: $\phi$ is of the form $\text{E} \varphi_1 \text{U} \varphi_2$

1. Add $\phi$ to labels to $s$ if $\varphi_2 \in \text{labels}(s)$
2. Add $\phi$ to labels to $s$ if
   - $\varphi \in \text{labels}(s')$
   - $(s, s') \in \rightarrow$
   - $\varphi_1 \in \text{labels}(s)$
3. Repeat step 2 as long as new labels can be added

Labelling algorithm

Case 6: $\phi$ is of the form $\text{EG} \varphi$

Basic idea

- look for loops on which $\varphi$ holds.
- look for paths on which $\varphi$ holds to those loops

Explore state space from states that satisfy $\varphi_2$ backwards, as long as states satisfy $\varphi_1$. The most challenging case
Labelling algorithm

Case 6: \( \phi \) is of the form \( EG \phi \)

Step 1: find loops on which \( \phi \) holds

Create graph \( M' = (S', \rightarrow', \mu') \) from \( M \) with
- \( S' \) are all states \( s \) with by removing all states \( s \in S \) in which \( \phi \in \text{labels}(s) \)
- update \( \rightarrow', \mu' \) accordingly

This is a graph, not a Kripke structure

Find nontrivial strongly connected components of \( M' \)
- A strongly connected component (SCC) \( C \) is
  - a maximal subgraph such that every node in \( C \) is reachable by every other node in \( C \) on a directed path that is contained entirely within \( C \).
- \( C \) is nontrivial iff either
  - it has more than one node or
  - it contains one node with a self loop

Use Tarjan’s algorithm to compute SCCs

Step 2: find paths on which \( \phi \) holds to SCCs

1. Add \( \phi \) to labels to \( s \in S' \) if \( s \) is in a SCC
2. Add \( \phi \) to labels to \( s \in S' \) if
   - \( \phi \in \text{labels}(s') \)
   - \( (s,s') \in \rightarrow' \)
   - \( \phi \in \text{labels}(s) \)
3. Repeat step 2 as long as new labels can be added

Lemma: \( M,s \models EG \phi \) iff

1. \( s \in S' \)
2. There exists a path in \( M' \) that leads from \( s \) to a nontrivial strongly connected component of \( M' \).

Intuition behind proof
- If there exists a path from \( s \) to a cycle and \( \phi \) holds in every state (by construction), then there exists an infinite path on which \( \phi \) holds
- If there exists an infinite path over finite states, then it must end in a cycle, i.e. a sub-graph of a SCC.
Labelling algorithm

Summary

- Start with the atomic propositions, and proceed with sub-formulae as follows
  1. If $\phi \in \text{AP}$ label $s$ if $\phi \in \mu(s)$
  2. If $\phi \equiv \neg \psi$ label all states not labelled $\phi$
  3. If $\phi \equiv \psi_1 \land \psi_2$ label all states labelled $\psi_1$ and $\psi_2$
  4. If $\phi = \text{EX} \psi$, if it has successor labelled $\phi$
  5. If $\phi = \text{EG} \psi$, explore state space from states that satisfy $\phi_1$ backwards, as long as states satisfy $\phi_1$
  6. If $\phi = \text{EG} \psi$ label states in SCCs of the graph, restricted to the states that satisfy $\phi$. Backtrack from those states.
Example
Start with the atomic propositions

Example
Go up one level

Example
Find nontrivial SCCs

Go up one level
Example

Backtrack

Example

Backtrack

Example

Backtrack

Example

Backtrack

Label all states not labelled $E\left[\text{true}\cup\text{EG}\left(pc_1 = 1 \land pc_2 = 1\right)\right]$$\Rightarrow M$ does not satisfy $AG\, AF\left\{\left(pc_1 = 1 \land pc_2 = 1\right)\right\}$
CTL Model checking

Complexity

- partitioning the states into strongly connected components is $O(|S|+|\rightarrow|)$
- exploring the transition relation has complexity $O(|S|+|\rightarrow|)$
- $n$ sub-formulas of the CTL formula $\phi$$\Rightarrow$ complexity is $O(|\phi| \cdot (|S|+|\rightarrow|))$

Fairness and Model Checking

Reminder

- weak fairness
  - if an event is continuously enabled, it will occur infinitely often
  - in LTL: $GF (\neg enabled \lor occurs)$
- strong fairness
  - if an event is infinitely often enabled it will occur infinitely often
  - in LTL: $GF enabled \Rightarrow GF occurs$

Weak/strong fairness can be expressed in LTL, however, not in CTL.

Fair CTL model checking

Given a strong CTL fairness constraint $\Psi_{fair} = GF \Psi_1 \Rightarrow GF \Psi_2$

with $\Psi_1$ and $\Psi_2$ CTL formulas.

The behaviour of $M$ is restricted to paths that are fair.

Fairness constraint is LTL formula over CTL state formulas!
Fair CTL model checking

Fair semantics for CTL state formulas

- \[ M,s \models p \text{ iff } p \in \mu(s) \]
- \[ M,s \models \neg \phi \text{ iff } M,s \not\models \phi \]
- \[ M,s \models \phi_1 \land \phi_2 \text{ iff } M,s \models \phi_1 \text{ and } M,s \models \phi_2 \]
- \[ M,s \models \forall \phi \text{ iff } \text{ for all fair paths } \pi \text{ starting in } s, M,\pi \models \phi \]
- \[ M,s \models \exists \phi \text{ iff there exists a fair path } \pi \text{ starting in } s \text{ such that } M,\pi \models \phi \]

Semantics for path formulas remain the same.

Fair CTL model checking

Given fairness constraint \( \Psi_{\text{fair}} = GF \Psi_1 \Rightarrow GF \Psi_2 \) and Kripke structure \( M=(S, \rightarrow, \mu) \):

Label all states that satisfy \( \Psi_1 \) and \( \Psi_2 \) with \( \Psi_1 \) and \( \Psi_2 \).

Use CTL model checking

Fair CTL model checking

Revisiting the cases

Given a CTL formula \( \phi \) in ENF deal with sub-formulae as follows:

1. If \( \phi \in \text{AP} \) label \( s \) if \( \phi \in \mu(s) \)
2. If \( \phi = \neg \psi \) label all states not labelled \( \psi \)
3. If \( \phi = \psi \land \psi \) label all states labelled \( \psi \) and \( \psi \)

The first three cases remain the same.

Fair CTL model checking

Case 4: \( \phi \) is of the form \( EX \psi \)

Add \( \phi \) to labels of \( s \) if \( \exists (s,s') \in \rightarrow \text{ such that } \psi \in \text{labels}(s') \) and \( M,s' \models E\Psi_{\text{fair}} \)

We use

\[ M,\pi \models \Psi_{\text{fair}} \text{ iff } \exists k \geq 0, M,\pi^k \models \Psi_{\text{fair}} \text{ iff } \forall k \geq 0, M,\pi^k \models \Psi_{\text{fair}} \]
Fair CTL model checking

Computing $M, s \models E \Psi_{fair}$

Basic idea

Find a path from $s$ to a cycle $s_0, \ldots, s_n$ such that either

- for all $0 \leq i \leq n \quad \Psi_1 \not\in \text{label}(s_i)$
- or there exist $0 \leq i \leq n \quad \Psi_2 \in \text{label}(s_i)$

Labelling

1. Label all states in SCCs $C$ of $M$ with $\Psi_{fair}$ if
   - there exists a state $s \in C$ s.t. $\Psi_2 \in \text{label}(s)$
   - or if the exists a SCC $D$ in $C'$, the restriction of $C$ to states with $\Psi_1 \not\in \text{label}(s)$

2. Backtrack from there, labelling states

3. Label states $EX\phi$ if they have a successor labelled $\phi$ and $\Psi_{fair}$

Case 5: $\phi$ is of the form $E \phi_1 U \phi_2$

1. Add $\phi$ to labels to $s$ if $\phi_2 \in \text{labels}(s)$
2. Add $\phi$ to labels to $s$ if
   - $\phi, \Psi_{fair} \in \text{labels}(s')$
   - $(s, s') \in \rightarrow$
   - $\phi \in \text{labels}(s)$
3. Repeat step 2 as long as new labels can be added

Computate the states that must be labelled $\Psi_{fair}$ as before

Case 6: $\phi$ is of the form $EG\phi$

1. Create graph $M' = (S', \rightarrow', \mu')$ from $M$ with
   - $S'$ are all states $s$ with by removing all states $s \in S$ in which $\phi \in \text{labels}(s)$ and update $\rightarrow$, $\mu'$ accordingly
2. Label all states in $M'$ that satisfy $E \Psi_{fair}$
Fair CTL model checking

Complexity

- For each several fairness constraints procedure has to be applied recursively
- For n sub-formulas of the CTL formula \( \varphi \), and k fairness constraints

\( \Rightarrow \text{complexity is } O(|\varphi| \times (|S|+|\rightarrow|) \times k) \)

Summary

- CTL model checking is
  - Linear in the size of the state space
  - Linear in the length of the formula
  - Linear in the number of fairness constraints

- Fairness constraints are few.
- Formulas are short.
- States explode!