Algorithmic Verification

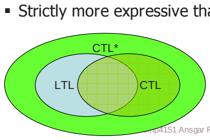
Comp4151
Lecture 4-A
Ansgar Fehnker

Overview

- 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.

Overview

- Specification
  - Linear Time Logic
    - Describes behaviour 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

Kripke strukture

temporal logic formula

$M \models \phi$

today:CTL

model, program

satisfies, implements, refines

property, specification

M statisfies ϕ iff all initial states satisfy ϕ

CTL model checking

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

CTL model checking

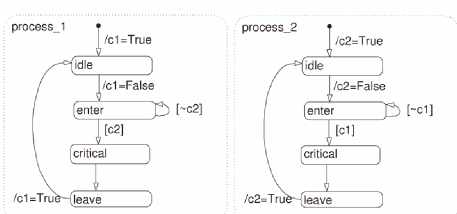
```
bool c1 := True
bool c2 := True

  While(True){               //process 1 |   While(True){               //process 2
0   c1 := False                          | 0   c2 := False
1   While(!c2){}             //busy wait | 1   While(!c1){}             //busy wait
2   critical section 1                   | 2   critical section 2
3   c1 := True}                          | 3   c2 := True}
```

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.

CTL model checking

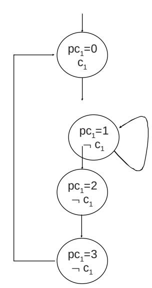


Mutual exclusion example:

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.

CTL model checking

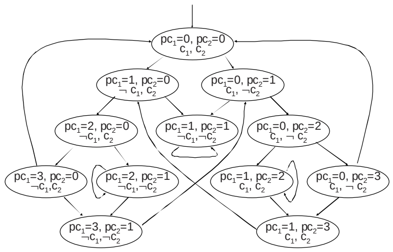
```
  While(True){               //process 1
0   c1 := False
1   While(!c2){}             //busy wait
2   critical section 1
3   c1 := True}
```



Atomic propositions:

- $pc_1=0$
- $pc_1=1$
- $pc_1=2$
- $pc_1=3$
- c_1

CTL model checking



Kripke structure

CTL model checking

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

CTL model checking

Syntax

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

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

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

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

CTL model checking

Example

$AG\ AF\ \neg(pc_1 = 1 \wedge pc_2 = 1)$

is equivalent to

$\neg\ E[\ \mathit{true}\ U\ EG\ (pc_1 = 1\ \wedge\ pc_2 = 1)\ ]$

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
  - For all sub-formulas label states that satisfy them

CTL model checking

For all sub-formulas label states that satisfy them

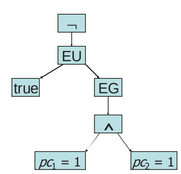
- Recursive bottom-up computation:
  - consider the parse-tree of ϕ
  - start with atomic propositions *p* in the leafs of the tree
    - for all states *s* if $p \in \mu(s)$ add *p* to the labels of *s*
  - go one level up in the tree and check sub-formula
    - if subformula is true in *s*, add it to *labels(s)*
  - proceed until the root of the tree is checked
- M |= ϕ if the initial state is labelled ϕ

CTL model checking

Example

$\neg\ E[\ \mathit{true}\ U\ EG\ (pc_1 = 1\ \wedge\ pc_2 = 1)\ ]$

Consider the parse tree



CTL Model checking

- Let *label(s)* the set of labels of state *s*
- Initially *label(s)={true}*
- Given a sub-formula ϕ in ENF there are six cases to consider

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

Labelling algorithm

Case 1: $\phi \in AP$

Add ϕ to labels of s if $\phi \in \mu(s)$

Labelling algorithm

Case 2: ϕ is of the form $\neg\varphi$

Add ϕ to labels of s if $\varphi \notin labels(s)$

Labelling algorithm

Case 3: ϕ is of the form $\phi_1 \wedge \phi_2$

Add ϕ to labels of s if $\phi_1, \phi_2 \in labels(s)$

Labelling algorithm

Case 4: ϕ is of the form EX φ

Add ϕ to labels of *s* if

$\exists$ (s,s') $\in \rightarrow$ such that $\varphi \in$ *labels* (*s'*)

Labelling algorithm

Case 5: ϕ is of the form E ϕ_1U ϕ_2

1. Add ϕ to labels to *s* if $\phi_2 \in$ *labels* (*s*)
2. Add ϕ to labels to *s* if
   - $\phi \in$ *labels* (*s'*)
   - (*s,s'*) $\in \rightarrow$
   - $\phi_1 \in$ *labels* (*s*)
3. Repeat step 2 as long as new labels can be added

Labelling algorithm

Case 5: ϕ is of the form E ϕ_1U ϕ_2

1. Add ϕ to labels to *s* if $\phi_2 \in$ *labels* (*s*)
2. Add ϕ to labels to *s* if
   - $\phi \in$ *labels* (*s'*)
   - (*s,s'*) $\in \rightarrow$
   - $\phi_1 \in$ *labels* (*s*)
3. Repeat step 2 as long as new labels can be added

Explore state space from states that satisfy ϕ_2 backwards, as long as states satisfy ϕ_1

Labelling algorithm

Case 6: ϕ is of the form EG φ

The most challenging case

Basic idea

- look for loops on which φ holds.
- look for paths on which φ holds to those loops

Labelling algorithm

Case 6: ϕ is of the form EG φ

Step 1: find loops on which φ holds

Create graph M' = (S', →', μ') from M with

- S' are all states s with by removing all states s ∈ S in which $\varphi \notin$ labels (*s*)
- update →', μ' accordingly

This is a graph, not a Kripke structure

Labelling algorithm

Case 6: ϕ is of the form EG φ

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

Labelling algorithm

Case 6: ϕ is of the form EG φ

Step 2: find paths on which φ holds to SCCs

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

Labelling algorithm

Case 6: ϕ is of the form EG φ

Lemma: M,s $\models$ EG φ iff

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

Intuition behind proof

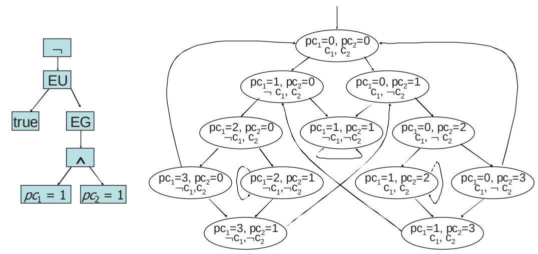
- If there exists a path from *s* to a cycle and φ holds in every state (by construction), then there exists an infinite path on which φ holds
- If there exists an infinite path over finite states, then it must end in a cycle, i.e. a sub-graph of a SCC.

Labelling algorithm

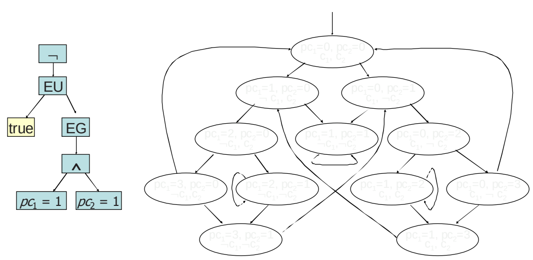
Summary

- Start with the atomic propositions, and proceed with sub-formulae as follows
  1. If $\phi \in AP$ label s if $\phi \in \mu(s)$
  2. If $\phi = \neg\varphi$ label all states not labelled φ
  3. If $\phi = \phi_1 \wedge \phi_2$ label all states labelled ϕ_1 and ϕ_2
  4. If $\phi = EX\ \varphi$, if it has successor labelled φ
  5. If $\phi = E\ \phi_1 U\ \phi_2$, explore state space from states that satisfy ϕ_2 backwards, as long as states satisfy ϕ_1
  6. If $\phi = EG\ \varphi$ label states in SCCs of the graph, restricted to the states that satisfy φ. Backtrack from those states.

Example

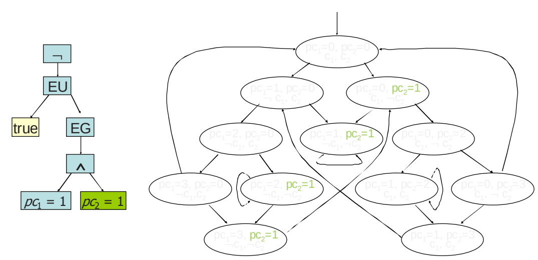


Example



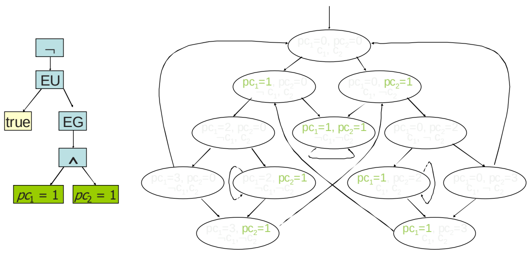
"True" is trivially in each set of labels

Example



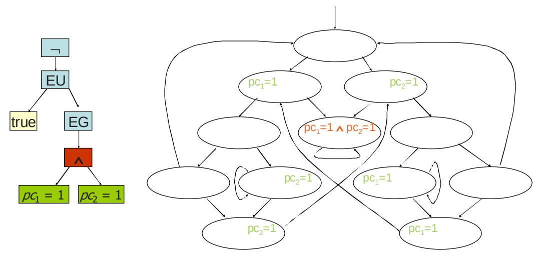
Start with the atomic propositions

Example



Start with the atomic propositions

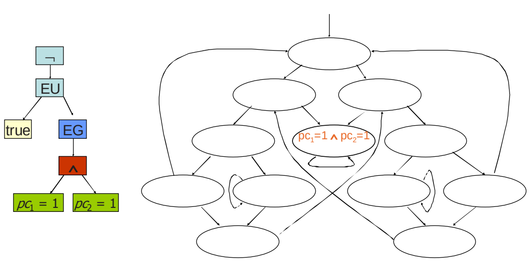
Example



Go up one level

Comp4151 Ansgar Fehnker

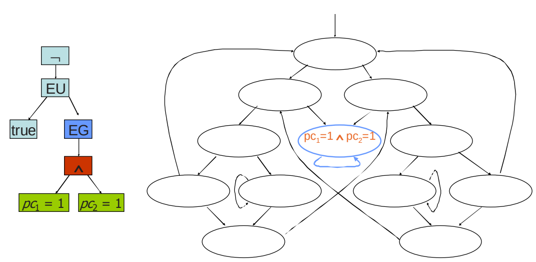
Example



Go up one level

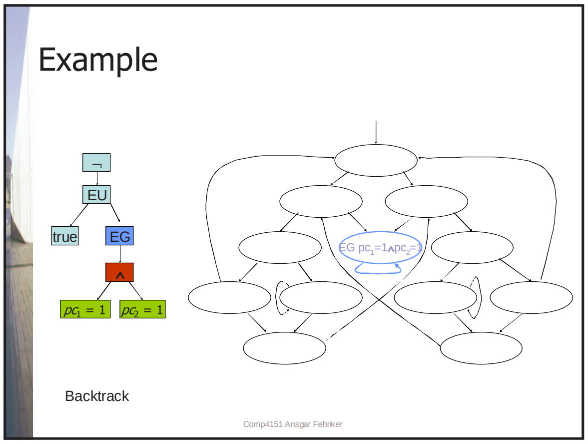
Comp4151 Ansgar Fehnker

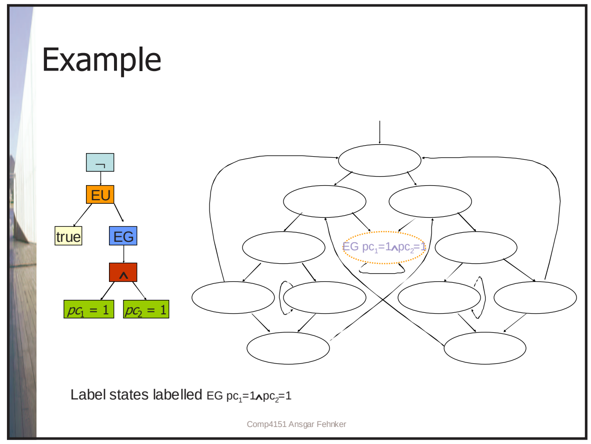
Example

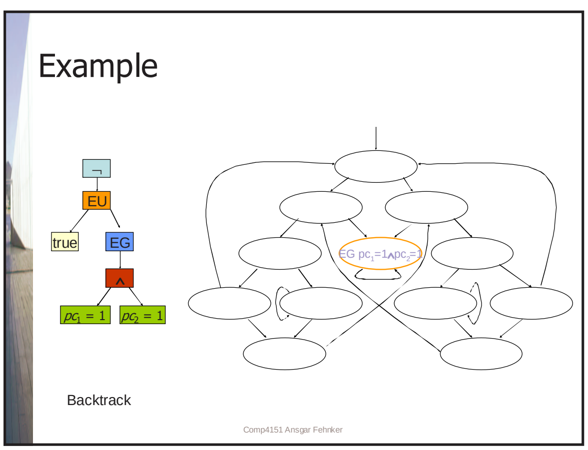


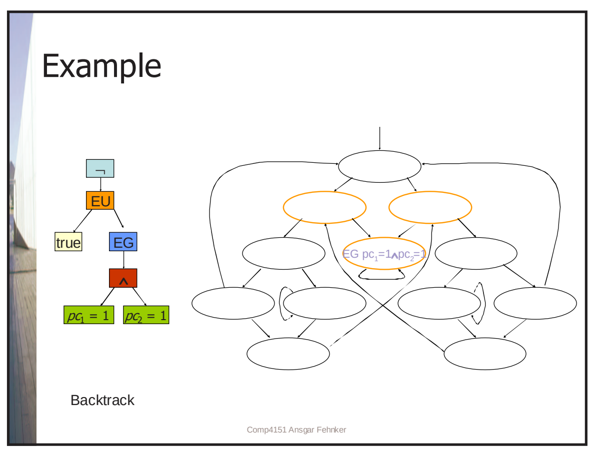
Find nontrivial SCCs

Comp4151 Ansgar Fehnker

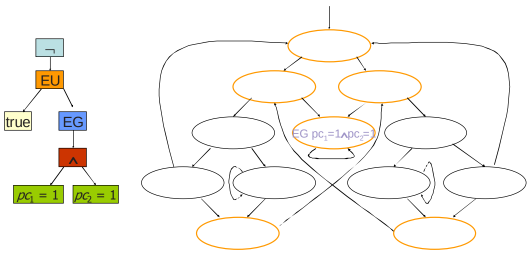

Example
¬
EU
true
EG
∧
pc1 = 1
pc2 = 1
EG pc1=1∧pc2=1
Backtrack
Comp4151 Ansgar Fehnker


Example
¬
EU
true
EG
∧
pc1 = 1
pc2 = 1
EG pc1=1∧pc2=1
Label states labelled EG pc1=1∧pc2=1
Comp4151 Ansgar Fehnker


Example
¬
EU
true
EG
∧
pc1 = 1
pc2 = 1
EG pc1=1∧pc2=1
Backtrack
Comp4151 Ansgar Fehnker

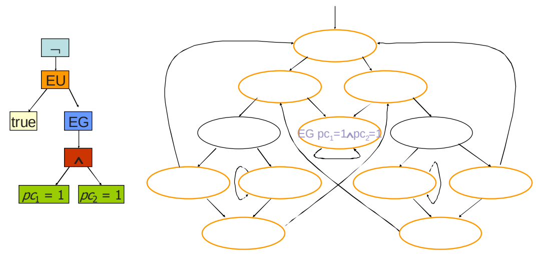

Example
¬
EU
true
EG
∧
pc1 = 1
pc2 = 1
EG pc1=1∧pc2=1
Backtrack
Comp4151 Ansgar Fehnker

Example



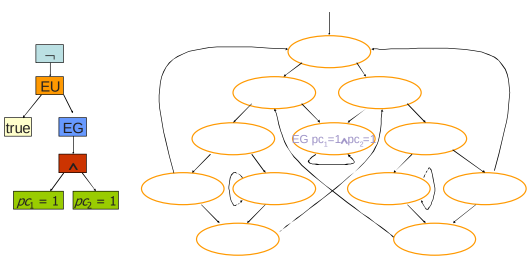
Backtrack

Example



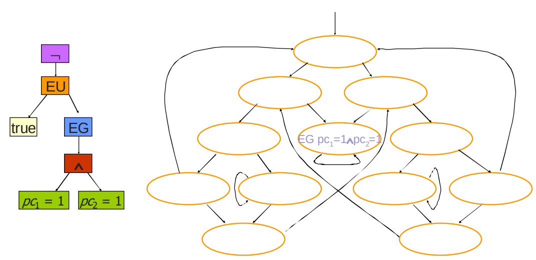
Backtrack

Example



Backtrack

Example



Label all states not labelled E[*true* U EG ($pc_1 = 1 \land pc_2 = 1$)]

⇒ M does not satisfy **AG AF ¬($pc_1 = 1 \land pc_2 = 1$)**

CTL Model checking

Complexity

- partitioning the states into strongly connected components is O(|S|+|→|))
- Exploring the transition relation has complexity O(|S|+|→|))
- n sub-formulas of the CTL formula ϕ

=> complexity is O(|ϕ| * (|S|+|→|))

Fairness and Model Checking

Reminder

weak fairness

if an event is continuously enabled, it will occur infinitely often

- in LTL: GF (¬enabled ∨ occurs)

strong fairness

if a event is infinitely often enabled it will occur infinitely often

- in LTL: GF enabled ⇒ GF occurs

Fairness and Model Checking

Reminder

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

in **LTL model checking** fairness can be added directly as an assumption

in **CTL model checking** fairness has to be build into the model checking algorithm

Fair CTL model checking

Given a strong CTL fairness constraint

Weak fairness analogously

$$\Psi_{fair} = GF\ \Psi_1 \Rightarrow GF\ \Psi_2$$

with Ψ_1 and Ψ_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 iff p$\in \mu$(s)
- M,s $\models \neg\phi$ iff not M,s $\models \phi$
- M,s $\models \phi_1 \wedge \phi_2$ iff M,s $\models \phi_1$ and M,s $\models \phi_2$
- M,s $\models$ Aϕ iff for all fair paths π starting in s, M,$\pi \models \phi$
- M,s $\models$ Eϕ iff there exists a fair path π starting in s, such that M,$\pi \models$ f

Semantics for path formulas remain the same.

Fair CTL model checking

Given fairness constraint Ψ_{fair} = GF $\Psi_1 \Rightarrow$ GF Ψ_2 and Kripke structure M=(S, s_0, $\rightarrow$, μ)

Label all states that satisfy Ψ_1 and Ψ_2 with Ψ_1 and Ψ_2

Use CTL model checking

Fair CTL model checking

Revisiting the cases

Given a CTL formula ϕ in ENF deal with sub-formulae as follows

1. If $\phi \in$ AP label s if $\phi \in \mu(s)$
2. If $\phi = \neg\varphi$ label all states not labelled φ
3. If $\phi = \phi_1 \wedge \phi_2$ label all states labelled ϕ_1 and ϕ_2

The first three cases remain the same

Fair CTL model checking

Case 4: ϕ is of the form EX φ

Add ϕ to labels of *s* if $\exists$ (s,s') $\in \rightarrow$ such that

$\varphi \in$ *labels*(*s'*) and M,s' $\models$ EΨ_{fair}

We use

M,$\pi \models \Psi_{fair}$ iff $\exists$k$\geq$0. M,$\pi^k \models \Psi_{fair}$ iff $\forall$k$\geq$0. M,$\pi^k \models \Psi_{fair}$

Fair CTL model checking

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

Basic idea

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

for all $0 \le i \le n$ $\Psi_1 \notin label(s_i)$ or
there exist $0 \le i \le n$ $\Psi_2 \in label(s_i)$

Fair CTL model checking

Labelling

1. Label all states in SCCs C of M with Ψ_{fair} if
   - there exists a $s \in C$ s.t. $\Psi_2 \in label(s)$ or
   - if the exists a SCC D in C', the restriction of C to states with $\Psi_1 \notin label(s)$
2. Backtrack from there, labelling states
3. Label states $EX\varphi$ if they have a successor labelled φ and Ψ_{fair}

Fair CTL model checking

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

1. Add ϕ to labels to *s* if $\phi_2 \in$ *labels* (*s*)
2. Add ϕ to labels to *s* if
   - $\phi, \Psi_{fair} \in$ *labels* (*s'*)
   - $(s,s') \in \rightarrow$
   - $\phi_1 \in$ *labels* (*s*)
3. Repeat step 2 as long as new labels can be added

Compute the states that must be labelled Ψ_{fair} as before

Fair CTL model checking

Case 6: ϕ is of the form $EG\ \varphi$

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 $\varphi \in$ *labels* (*s*) and update $\rightarrow'$, μ' 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 ϕ, and k fairness constraints

=> complexity is $O(|\phi| * (|S|+|\rightarrow|)* k)$

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 !