COL750: Foundations of Automatic Verification (Jul-Dec 2024)

CTL Model Checking and BDDs[1]

Kumar Madhukar

madhukar@cse.iitd.ac.in

September 5th

[1] reusing slides created by Prof. Jacques Fleuriot, University of Edinburgh (https://homepages.inf.ed.ac.uk/jdf/)

EX

function $\texttt{SAT}_{\texttt{EX}}\,(\phi)$
/* determines the set of states satisfying $\mathrm{EX}\,\phi$ */
local var X, Y
begin
    $X := \texttt{SAT}\,(\phi)$;
    $Y := \mathrm{pre}_{\exists}(X)$;
    return Y
end

EU

```
function SAT_EU (φ, ψ)
 /* determines the set of states satisfying E[φ U ψ] */
local var W, X, Y
begin
    W := SAT (φ);
    X := S;
    Y := SAT (ψ);
    repeat until X = Y
    begin
        X := Y;
        Y := Y ∪ (W ∩ pre_∃(Y))
    end
    return Y
end
```

EG

```
function SAT_EG (φ)
/* determines the set of states satisfying EG φ */
local var X, Y
begin
    Y := SAT (φ);
    X := ∅;
    repeat until X = Y
    begin
        X := Y;
        Y := Y ∩ pre_∃(Y)
    end
    return Y
end
```

CTL Model Checking with Fairness

CTL Model Checking with Fairness

- recall the mutex example, where processes were allowed to stay in their critical section as long as required
- this can lead to violation of the liveness constraint AG $(t_1 \rightarrow \text{AF } c_1)$
- we would like to ignore such paths (assuming that the processes would eventually exit from its critical section after some finite time)

CTL Model Checking with Fairness

- recall the mutex example, where processes were allowed to stay in their critical section as long as required
- this can lead to violation of the liveness constraint AG $(t_1 \rightarrow \text{AF } c_1)$
- we would like to ignore such paths (assuming that the processes would eventually exit from its critical section after some finite time)
- In LTL, we could handle this by saying GF $\neg c_2 \rightarrow \phi$

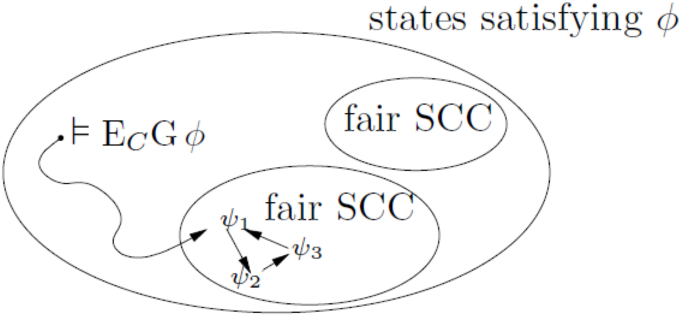
CTL Model Checking with Fairness

- CTL does not allow us to pick fair paths
- NuSMV allowed us to write FAIRNESS constraints
- NuSMV can handle only simple fairness constraints (of the form: ϕ is true infinitely often)
- fairness constraints may be more complex (e.g. if ϕ is true infinitely often, then ψ is true infinitely often)

Handling Simple Fairness

- Let $C := \{\psi_1, \psi_2, \ldots, \psi_n\}$ be n fairness constraints
- A computational path is called fair wrt these if every ψ_i is true infinitely often along that path
- Let A_C and E_C denote the operators A and E restricted to fair paths
- $E_C U$, $E_C X$, and $E_C G$ form an adequate set
- We need to handle only $E_C G$

Handing $E_C G$


states satisfying ϕ
fair SCC
$\models \mathrm{E}_C \mathrm{G}\, \phi$
fair SCC
ψ_1
ψ_2
ψ_3

State-space Explosion

- abstraction, decomposition, induction
- efficient data structures (binary decision diagrams)

Boolean functions

- an important descriptive formalism for many hardware and software systems
- efficient representation is desirable
- a boolean function of n arguments is a function from $\{0,1\}^n$ to $\{0,1\}$
- truth tables and propositional formulas are two different representations of boolean functions
- we may also represent them by subclasses of propositional formulas (e.g. CNF, DNF)
- different representations have different advantages and disadvantages

Binary Decision Diagrams

- was invented in the 1990s
- enabled the first practical SAT solver
- modern SAT solvers use CDCL

Thank you!