Algorithmic Verification

Comp4151
Lecture 9-B
Ansgar Fehnker

Overview

Model checking Approaches

- Explicit State Model Checking
  - Combat the state explosion problem by reduction techniques such as partial order or symmetry reduction
  - Common tool: Spin
- Symbolic Model Checking
  - Represents set of states and transition relation as BDD
  - Use fast and efficient BDD algorithms
  - Common tool: NuSMV

Overview

Model checking Approaches (cont)

- Bounded Model Checking
  - Relies on fast and efficient SAT-solvers
  - Transforms the bounded model checking problem to a satisfiability problem
  - BMC allows for tailored optimizations of SAT procedure
- Counterexample Guided Abstraction Refinement
  - Model check a small abstraction rather than full model
  - Refine the abstraction, if necessary, automatically

Abstraction

Existential Abstraction

A transition system is a Kripke structure without the assumption that R total

Definition

Given a transition system $M=(S, S_0, R, L)$, and a surjective (many-to-one) mapping $h: S \rightarrow \acute{S}$.

The (minimal) existential abstraction is a transition system M' with

- state space $\acute{S}$
- initial states $\acute{S}_0 = \{ h(s) \mid s \in S_0 \}$
- transition relation $\acute{R} = \{ (h(s_0),h(s_1)) \mid (s_0,s_1) \in R\}$
- and labeling $\acute{L}(\acute{s}) = \cup_{h(s)=\acute{s}} L(s)$

Abstraction

Existential Abstraction

Definition

Given a transition system M=(S, S_0, R, L), and a surjective (many-to-one) mapping h: S → Ś.

*also called the **concrete** model*

maps many states to a few states

The (minimal) existential abstraction is a transition system M' with

- state space Ś — *smaller than S*
- initial states $Ś_0$ ={ h(s) | s ∈ S_0 } — *initial states mapped to abstract initial states*
- transition relation Ŕ = { (h(s_0),h(s_1))| (s_0,s_1) ∈ R} — *transition between abstract states if there **exists** corresponding transition in M*
- and labeling Ĺ (ś)= $\cup_{h(s)=ś}$ L(s) — *union of all labels in corresponding states*

Abstraction

Types of abstraction

Predicate abstraction

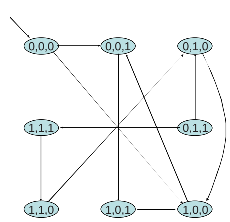
Given a set of predicates $p_0,\ldots,p_{n-1}$ the abstract state is a boolean vector $(b_0,\ldots,b_{n-1})$, such that state s is mapped to an abstract state iff

$$b_i \Leftrightarrow s|= p_i$$

Abstraction

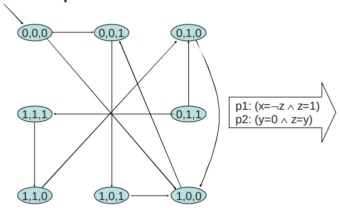
Example

```
variables
   bool x=0,y=0,z=0;

transitions
   pre:   x=0;
   post:  x'=1, y'=y∧z;

   pre:   y=z;
   post:  x'=y∧x, z=¬z;

   pre:   x=1,y=¬z;
   post:  x'=¬y, z'=¬x;

assert (x=0 ∨ y=0)
```



Abstraction

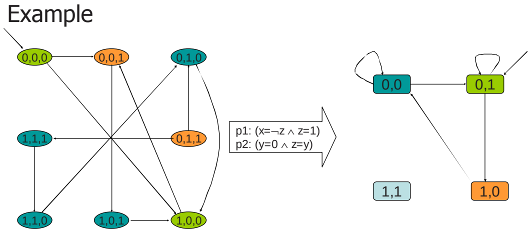
Example



Given predicates p1 and p2, the abstract state is a tuple (b_0,b_1)

Abstraction

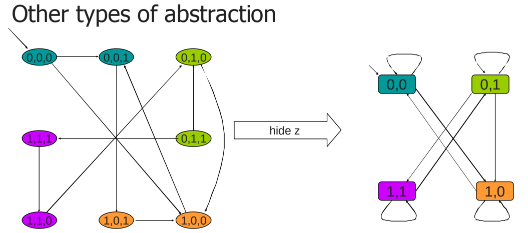
Example



Given predicates p1 and p2, the abstract state is a tuple (b_0,b_1)

Abstraction

Other types of abstraction



Abstraction by hiding variables

Abstraction

Example by Daniel Kroening

Abstraction as Boolean program

```
int main() {
  int i;

  i=0;

  while(even(i))
    i++;
}
```

+

$p_1 \Leftrightarrow i=0$
$p_2 \Leftrightarrow even(i)$

=

```
void main() {
  bool p1, p2;

  p1=TRUE;
  p2=TRUE;

  while(p2)
  {
    p1=p1?FALSE:nondet();
    p2=!p2;
  }
}
```

C program **Predicates** **Boolean program**

[Graf, Saidi '97]
[Ball, Rajamani '00]

Abstraction

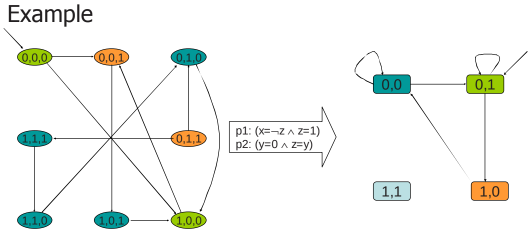
Consistent labelling

Definition

Labeling Ĺ is *consistent* with h if L(s) = Ĺ (h(s)) for all s.

Abstraction

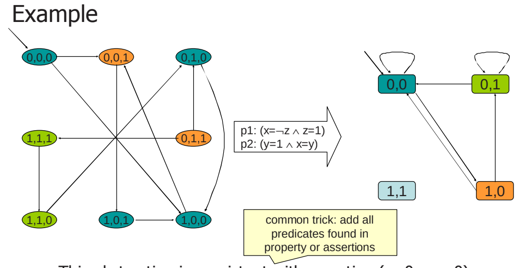
Example



This abstraction is not consistent with assertion (x=0 ∨ y=0)

Abstraction

Example



This abstraction is consistent with assertion (x=0 ∨ y=0)

Abstraction

Preservation Theorem

Theorem

Let M' be a existential abstraction of M, for abstraction function h. Assume that the labeling of M' is consistent with h. Given an ACTL property ϕ we have

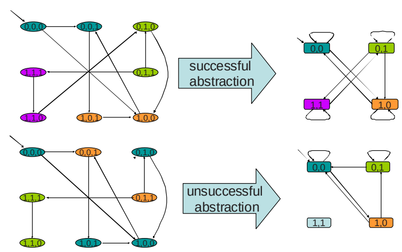
$$M' \models f \Rightarrow M \models \phi$$

If we can show by model checking that M' is correct, we don't need to check M.

We hope that M' is much smaller than M, thus easier to model check.

Abstraction

Show AG ¬ (x=1 ∧ y=1)



Abstraction Refinement

Refinement

Definition

Given a transition system M and an abstraction function h from S to Ś. An abstraction function h′ from S to S̋ is a *refinement* if

$$h'(s_0)=h'(s_1) \Rightarrow h(s_0)=h(s_1)$$

What does this mean?

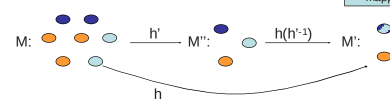
Abstraction Refinement

Series of Abstractions

Given and transition system M, abstraction function h, and a refinement h′ of h

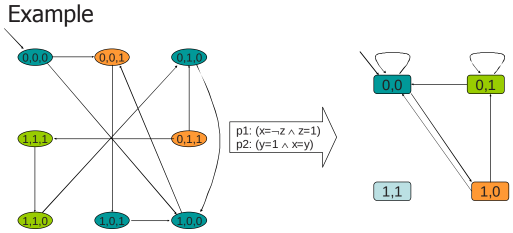
- M′ is an abstraction of M w.r.t h
- M″ is also an abstraction of M, but w.r.t h′
- And also: M′ is an abstraction of M″

it is possible to show that there exists a surjective mapping from S″ to S′



Abstraction Refinement

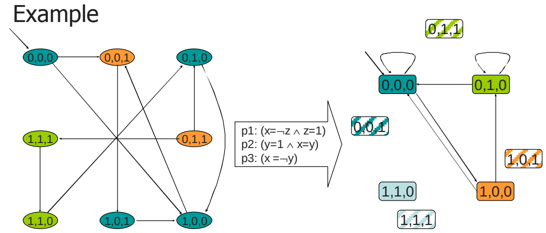
Example



Refinement by introduction of a new predicate

Abstraction Refinement

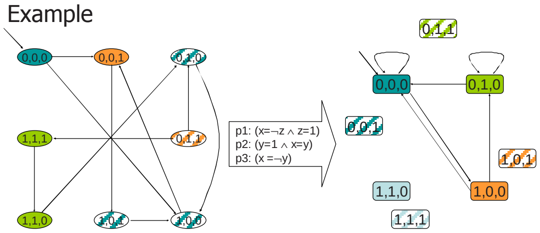
Example



Refinement by introduction of a new predicate

Abstraction Refinement

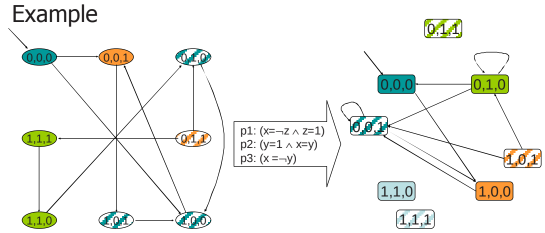
Example



Update the abstraction function

Abstraction Refinement

Example



Update of the transition relation

Abstraction Refinement

Series of Abstraction

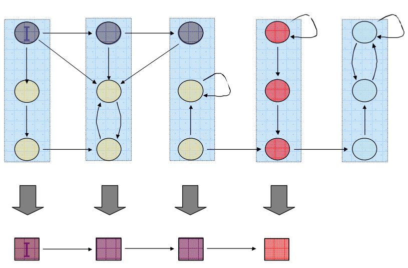
1. Construct automatically a series of abstractions M′, M″, M‴,… such that for some *n* $M^n \models \phi$
2. If $M^i \not\models \phi$ use the abstract counterexample to obtain information about how to refine M^i .
3. Check if the abstract counterexample of M^i corresponds to a real one in M. Then $M \not\models \phi$

Spurious Counterexample

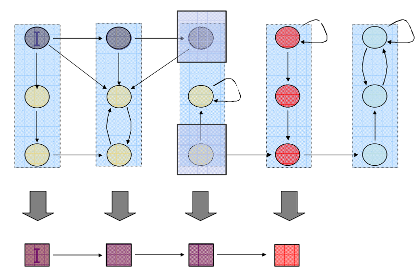
Definition

An abstract counterexample ($\acute{s}_0,\ldots,\acute{s}_n$) of M′ is *spurious,* if there exists no path ($s_0,\ldots,s_n$) in concrete model M, such that $h(s_i)=s'_i$, for all i.

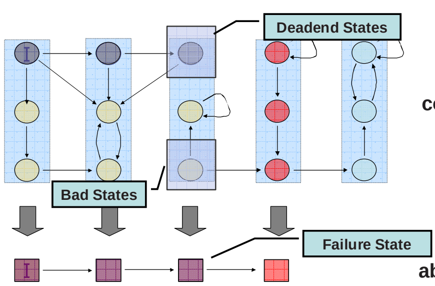
Spurious Counterexample



Spurious Counterexample

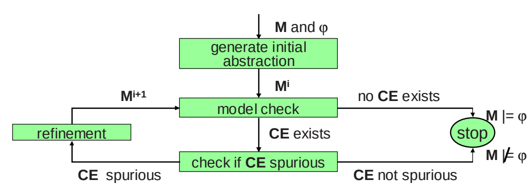


Spurious Counterexample



Automatic Abstraction Refinement

Refinement Loop



- We know how to model check (symbolically or explicit).
- But how to check a counterexample and how to refine?

Automatic Abstraction Refinement

Checking Counterexamples

Automatic Theorem Proving

- Given an abstract counterexample $(\acute{s}_0,\dots,\acute{s}_n)$ of M′
- Use automatic theorem prover to show that there exists no series of states $(s_0,\dots,s_n)$ such that
  - $s_0 \in S_0$
  - $(s_i,s_{i+1}) \in R$ for all i
  - $h(s_i)= \acute{s}_i$ for all i

Automatic Abstraction Refinement

Checking Counterexamples

SAT solving

- Given an abstract counterexample $(\acute{s}_0,\dots,\acute{s}_n)$ of M′
- There exists no corresponding concrete path by if the following is unsatisfiable

$$\Omega = I(s_0) \wedge \bigwedge_{i=0}^{n-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{n-1} h(s_i) = \acute{s}_i$$

- A satisfying assignment gives a real counterexample.
- If we find a satisfying assignment, then $M \not\models \phi$

Abstraction Refinement

Refining the abstraction

Automatic Theorem Prover

- Use predicates found by the theorem prover
- A lot of effort in finding the right predicates (small and useful)
- Details exceed scope of this lecture

Abstraction Refinement

Refining the abstraction

SAT-solving

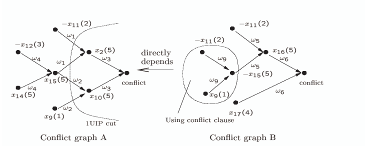
- The conflict clauses show why there exist no counterexample in M
- Use predicates found in conflict clauses, or
- Make variable visible that appear (a lot) in conflict clauses
- Reducing the set of relevant clauses by analysis of the conflict dependency graph.

Abstraction Refinement

Refining the abstraction

Conflict Dependency?

- A conflict clause A may appear in the implication graph of a second conflict B
- We then say that B depends directly on A

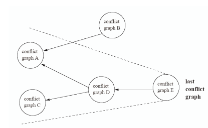


Abstraction Refinement

Refining the abstraction

Conflict dependency graph

- Build a conflict dependency graph



- Use only clauses on which the last conflict depends for refinement

Example

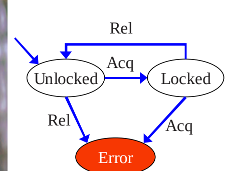
Example by Daniel Kroening

Driver verification with SLAM

- Microsoft blames most Windows crashes on third party device drivers
- SLAM: Tool to automatically check device drivers for certain errors
- Specification in SLIC: Finite state language for stating rules
  - monitors behavior of C code
  - temporal safety properties – similar to what SPIN does
  - familiar C syntax

Example

Locking Rule




```
state {
  enum {Locked,Unlocked}
     s = Unlocked;
}

KeAcquireSpinLock.entry {
  if (s==Locked) abort;
  else s = Locked;
}

KeReleaseSpinLock.entry {
  if (s==Unlocked) abort;
  else s = Unlocked;
}
```

Locking Rule in SLIC

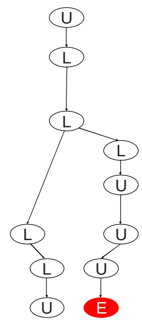
Example

Does this code obey the locking rule?

```
do {
  KeAcquireSpinLock();

  nPacketsOld = nPackets;

  if(request){
      request = request->Next;
      KeReleaseSpinLock();
      nPackets++;
  }
} while (nPackets != nPacketsOld);

KeReleaseSpinLock();
```


Example

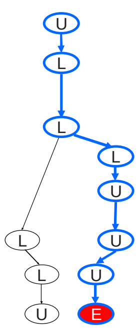
Model checking boolean program (bebop)



```
do {
  KeAcquireSpinLock();


  if(*){

      KeReleaseSpinLock();

  }
} while (*);

KeReleaseSpinLock();
```


Example

Is error path feasible in C program? (newton)

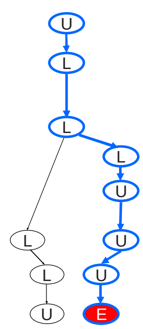


```
do {
  KeAcquireSpinLock();

  nPacketsOld = nPackets;

  if(request){
      request = request->Next;
      KeReleaseSpinLock();
      nPackets++;
  }
} while (nPackets != nPacketsOld);

KeReleaseSpinLock();
```

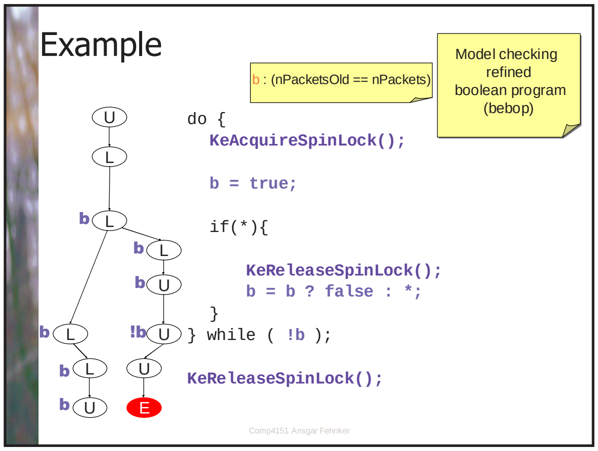

Example

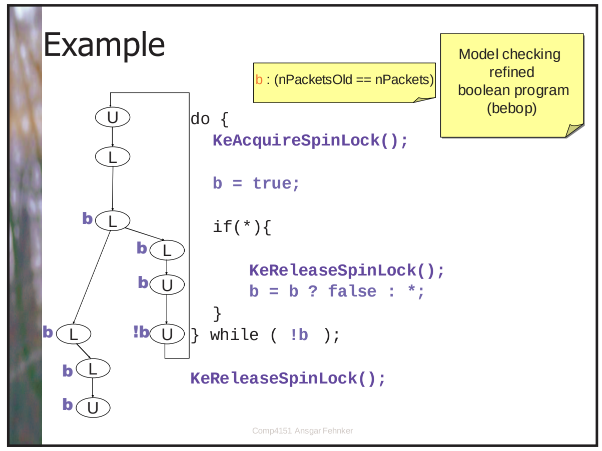
b : (nPacketsOld == nPackets)

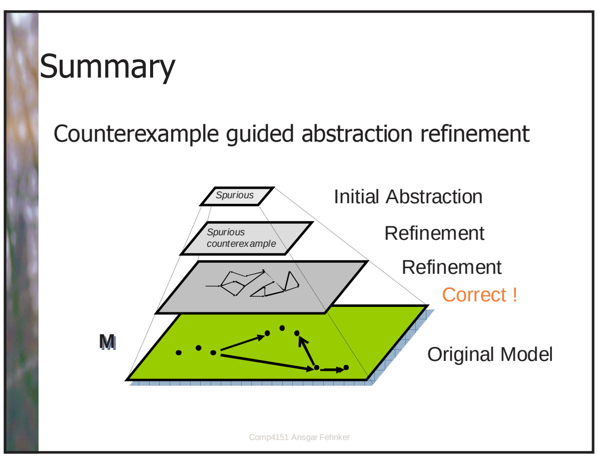
Add new predicate to boolean program (c2bp)

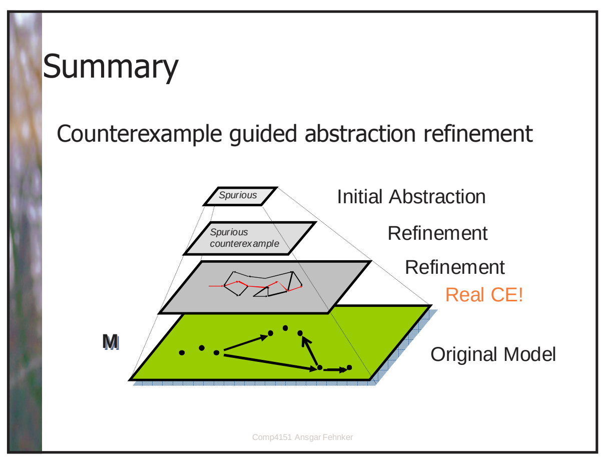


```
do {
  KeAcquireSpinLock();

  nPacketsOld = nPackets; b = true;

  if(request){
      request = request->Next;
      KeReleaseSpinLock();
      nPackets++; b = b ? false : *;
  }
} while (nPackets != nPacketsOld);  !b

KeReleaseSpinLock();
```


Example
b : (nPacketsOld == nPackets)
Model checking refined boolean program (bebop)
do {
KeAcquireSpinLock();
b = true;
if(*){
KeReleaseSpinLock();
b = b ? false : *;
}
} while (!b);
KeReleaseSpinLock();
U
L
b L
b L
b U
!b U
b L
b L
b U
U
E
Comp4151 Ansgar Fehnker


Example
b : (nPacketsOld == nPackets)
Model checking refined boolean program (bebop)
do {
KeAcquireSpinLock();
b = true;
if(*){
KeReleaseSpinLock();
b = b ? false : *;
}
} while (!b);
KeReleaseSpinLock();
U
L
b L
b L
b U
!b U
b L
b L
b U
Comp4151 Ansgar Fehnker


Summary
Counterexample guided abstraction refinement
Spurious
Initial Abstraction
Spurious counterexample
Refinement
Refinement
Correct !
M
Original Model
Comp4151 Ansgar Fehnker


Summary
Counterexample guided abstraction refinement
Spurious
Initial Abstraction
Spurious counterexample
Refinement
Refinement
Real CE!
M
Original Model
Comp4151 Ansgar Fehnker

Conclusions

Predicate abstraction and abstraction refinement have become a standard technique in software verification

(C programs) SLAM '00
  - Mircrosoft Research
  - Abstract C programs to Boolean programs
- (C programs) BLAST
  - Berkeley and Los Angeles
  - On-the-fly Predicate Abstraction and proof-based CE analysis
- (C programs) MAGIC
  - CMU
  - SAT-based CE analysis
- (Java programs) ESC/Java, Bandera, …

Conclusion

Next week

Static Analysis

What else can you do to check the correctness of software?