Predicate Abstraction
Predicate Abstraction

- Extract a finite state model from an infinite state system
- Used to prove assertions or safety properties
- Successfully applied for verification of C programs
  - SLAM (used in windows device driver verification)
  - MAGIC, BLAST, F-Soft
Example for Predicate Abstraction

C program

```c
int main() {
    int i;
    i=0;
    while(even(i))
        i++;
}
```

Predicates

```plaintext
\[
\begin{align*}
p_1 & \iff i=0 \\
p_2 & \iff \text{even}(i)
\end{align*}
\]
```

Boolean program

```c
void main() {
    bool p1, p2;
    p1=TRUE;
    p2=TRUE;
    while(p2)
        {
            p1=p1?FALSE:nondet();
            p2=!p2;
        }
}
```

[Ball, Rajamani '01]

[Graf, Saidi '97]
Computing Predicate Abstraction

- How to get predicates for checking a given property?
- How do we compute the abstraction?
- Predicate Abstraction is an over-approximation
- How to refine coarse abstractions
Counterexample Guided Abstraction Refinement loop

1. **C Program**
2. **Initial Abstraction**
   - Abstract model
3. **Verification**
   - Model Checker
     - Property holds
     - No error or bug found
   - Refinement
     - Simulation successful
     - Bug found
4. **Spurious counterexample**

---

**Abstract**

- Program
- Model Checker
- Simulator
- Refinement
- Spurious counterexample
Example ( ) {
1: do {
   lock();
   old = new;
   q = q->next;
2:   if (q != NULL){
3:     q->data = new;
4:       unlock();
5:       new ++;
   }
4: } while(new != old);
5: unlock ();
6: return;
}
What a program really is...

**State**

```
3: unlock();
new++;

pc  → 3
lock  → 4
old  → 5
new  → 6
q  → 0x133a
```

**Transition**

```
Example () {
1: do{
    lock();
    old = new;
    q = q->next;
2:    if (q != NULL){
3:        q->data = new;
        unlock();
        new ++;
    }
4:  } while(new != old);
5:  unlock ();
  return;}
```
The Safety Verification Problem

Is there a path from an initial to an error state?

**Problem:** Infinite state graph

**Solution:** Set of states’ logical formula
Idea 1: Predicate Abstraction

- **Predicates** on program state: 
  - `lock`
  - `old = new`

- States satisfying **same** predicates are **equivalent**
  - Merged into one abstract state

- **#abstract states is finite**
Abstract States and Transitions

```
3: unlock();
new++; 4:
```

State

```
lock old=new
old = 5
new = 5
q = 0x133a
pc = 4
lock = 5
old = 5
new = 6
q = 0x133a
```

lock
old=new

! lock
! old=new
Abstraction

Existential Approximation

3: unlock();
   new++;
4: } ...

pc  → 3
lock → 5
old  → 5
new  → 5
q    → 0x133a

pc  → 4
lock →
old  → 5
new  → 6
q    → 0x133a

lock
old=new

! lock
! old=new
Abstraction

State

3: unlock();
    new++;
4: } ...

lock
old=new

! lock
! old=new
Analyze Abstraction

Analyze finite graph

Over Approximate:
Safe => System Safe

Problem
Spurious counterexamples
Idea 2: Counterex.-Guided Refinement

Solution
Use spurious counterexamples to refine abstraction!
Idea 2: Counterex. - Guided Refinement

Solution
Use spurious counterexamples to refine abstraction

1. Add predicates to distinguish states across cut
2. Build refined abstraction
Iterative Abstraction-Refinement

1. Add predicates to distinguish states across cut
2. Build refined abstraction - eliminates counterexample
3. Repeat search Till real counterexample or system proved safe

Solution
Use spurious counterexamples to refine abstraction

[Kurshan et al 93] [Clarke et al 00] [Ball-Rajamani 01]
Example () {
    do {
        lock();
        old = new;
        q = q->next;
        if (q != NULL) {
            q->data = new;
            unlock();
            new ++;
        }
    } while (new != old);
    unlock();
}
Example ( ) {
1:   do{
    lock();
    old = new;
    q = q->next;
2:     if (q != NULL){
3:       q->data = new;
        unlock();
      new ++;
    }
4: }while(new != old);
5: }unlock ();
}
Example () {
1: do{
    lock();
    old = new;
    q = q->next;
2:    if (q != NULL){
3:      q->data = new;
        unlock();
        new ++;
    }
4:} while(new != old);
5: unlock();
}

Predicates: LOCK
Example ( ) {
1: do{
   lock();
   old = new;
   q = q->next;
2:   if (q != NULL){
3:     q->data = new;
      unlock();
      new ++;
   }
4:}while(new != old);
5: unlock ();
}
Build-and-Search

Example ( ) {
1: do{
   lock();
   old = new;
   q = q->next;
2:   if (q != NULL){
3:     q->data = new;
        unlock();
        new ++;
   }
4:}while(new != old);
5: unlock ();
}
Example () {
1:   do{
     lock();
     old = new;
     q = q->next;
2:     if (q != NULL){
3:       q->data = new;
       unlock();
       new ++;
    }
4:   }while(new != old);
5:   unlock();
}
Analyse Counterexample

Example ( ) {
1: do{
    lock();
    old = new;
    q = q->next;
2:   if (q != NULL){
3:     q->data = new;
        unlock();
        new ++;
    }
4:}while(new != old);
5: unlock ();
}

Predicates: \textit{LOCK}
Analyze Counterexample

Example ( ) {
1:   do{
      lock();
      old = new;
      q = q->next;
    2:   if (q != NULL){
3:     q->data = new;
       unlock();
       new ++;
    } 4:}while(new != old);
5:     unlock ();
}

Predicates: LOCK

Example ( ) {
1:   do{
      lock();
      old = new;
      q = q->next;
    2:   if (q != NULL){
3:     q->data = new;
       unlock();
       new ++;
7: } 4:}while(new != old);
5:     unlock ();
}

old = new
new++
Inconsistent
[new==old]
new == old
Repeat Build-and-Search

Predicates: \textit{LOCK, new==old}
Repeat Build-and-Search

Example ( ) {
1:   do{
    lock();
    old = new;
    q = q->next;
    if (q != NULL){
      q->data = new;
      unlock();
      new ++;
    }
  }while(new != old);
5:   unlock ();
}
Repeat Build-and-Search

Example ( ) {
    do{
        lock();
        old = new;
        q = q->next;
        if (q != NULL){
            q->data = new;
            unlock();
            new ++;
        }
    }while(new != old);
    unlock ();
}
**Repeat Build-and-Search**

**Example ( ) {**

1: `do{`
   `lock();`
   `old = new;`
   `q = q->next;`
2: `if (q != NULL){`
3:   `q->data = new;`
   `unlock();`
   `new ++;`
4: `}`
5: `while(new != old);`

6: `unlock ();`**

**Predicates:** \( \text{LOCK, new==old} \)
Repeat Build-and-Search

Example ( ) {
    do{
        lock();
        old = new;
        q = q->next;
        if (q != NULL){
            q->data = new;
            unlock();
            new ++;
        }
    }while(new != old);
    unlock ();
}

Predicates: LOCK, new==old
Repeat Build-and-Search

Predicates: \( \text{LOCK, new==old} \)

Example ( ) {
1: do{
    lock();
    old = new;
    q = q->next;
2:   if (q != NULL){
3:     q->data = new;
        unlock();
        new ++;
    }
4:}while(new != old);
5: unlock ();
}
Another Example

C program

1: x = ctr;
2: y = ctr + 1;
3: if (x = i-1){
4:   if (y != i){
      ERROR:
    }
  }

No predicates available currently

1: skip;
2: skip;
3: if (*){
4:   if (*){
      ERROR:
    }
  }
Checking the abstract model

Is ERROR reachable?

Abstract model has a path leading to error state

1: skip;
2: skip;
3: if (*){
4:   if (*){
            ERROR:
             }
    }
}
yes
Does this correspond to a real bug?

### Concrete trace

1. `x = ctr;`
2. `y = ctr + 1;`
3. `assume(x == i-1)`
4. `assume (y != i)`

### Check using a SAT solver

Not possible

### Does this correspond to a real bug?

1. `skip;`
2. `skip;`
3. `if (*){`
4. `if (*){`
   
   ERROR:
   ```
   }
   }
   ```
1: x = ctr;
2: y = ctr + 1;
3: assume(x == i-1)
4: assume (y != i)

1: skip;
2: skip;
3: if (*){
4:   if (*){
     ERROR:
   }
}

Spurious Counterexample

Initial abstraction
Refinement

1: x = ctr;
2: y = ctr + 1;
3: assume(x == i-1)
4: assume(y != i)

1: skip;
2: skip;
3: if (*){
4: if (b0){
    ERROR:
    }
  }
boolean b0 : y != i
Refinement

1: \( x = \text{ctr}; \)
2: \( y = \text{ctr} + 1; \)
3: `assume(x == i-1)`
4: `assume(y != i)`

```java
1: skip;
2: skip;
3: if (b1){
4:   if (b0){
ERROR:
   }
}
```

**boolean b0 :** \( y \neq i \)

**boolean b1 :** \( x == i-1 \)
Weakest Preconditions

\[ WP(P, OP) \]

Weakest formula \( P' \) s.t.
if \( P' \) is true before \( OP \)
then \( P \) is true after \( OP \)
Weakest Preconditions

\( WP(P, OP) \)

Weakest formula \( P' \) s.t.
if \( P' \) is true before \( OP \)
then \( P \) is true after \( OP \)

Assign
\[ x = e \]

\[ P[e/x] \]

\[ new = old \]

\[ new+1 = old \]

\[ new = new+1 \]
Weakest precondition of \( y \neq i \)

1. \( x = \text{ctr}; \)
2. \( y = \text{ctr} + 1; \)
3. \( \text{assume}(x == i-1) \)
4. \( \text{assume}(y != i) \)

1. \( \text{skip}; \)
2. \( b0 = b2; \)
3. \( \text{if}(b1)\{ \)
4. \( \text{if}(b0)\{ \)
   
   \text{ERROR:} 

\}
\}

boolean \( b0 : y != i \)

boolean \( b1 : x == i-1 \)
Refinement

1. \( x = \text{ctr}; \)
2. \( y = \text{ctr} + 1; \)
3. \( \text{assume}(x == i-1) \)
4. \( \text{assume}(y != i) \)

Boolean variables:
- \( b0 : y != i \)
- \( b1 : x == i-1 \)
- \( b2 : \text{ctr} + 1 != i \)
- \( b3 : \text{ctr} == i - 1 \)
Refinement

What about initial values of b2 and b3?

b2 = 1, b3 = 0

b2 = 0, b3 = 1

b2 and b3 are mutually exclusive.

boolean b0 : y != i

boolean b1 : x == i - 1

boolean b2 : ctr + 1 != i

boolean b3 : ctr == i - 1

1: b1 = b3;
2: b0 = b2;
3: if (b1){
   if (b0){
      ERROR:
   }
}

So system is safe!
Tools for Predicate Abstraction of C

- SLAM at Microsoft
  - Used for verifying correct sequencing of function calls in Windows device drivers

- MAGIC at CMU
  - Allows verification of concurrent C programs
  - Found bugs in MicroC OS

- BLAST at Berkeley
  - Lazy abstraction, interpolation

- SATABS at CMU
  - Computes predicate abstraction using SAT
  - Can handle pointer arithmetic, bit-vectors

- F-Soft at NEC Labs
  - Localization, register sharing