Predicate Abstraction for Reactive Synthesis

Adam Walker, Leonid Ryzhyk
Automatic Device Driver Synthesis

- A three-year project (2013-2016)
- Funded by a $1.2M grant from Intel
Motivation

• Termite device driver synthesis project (OSDI '14)
• Driver vs (OS || device)
• Upwards of $2^{1000}$ states
• Predicate abstraction to capture relationships between variables
Background

- $G = \langle S, L, I, \tau_1, \tau_2, \delta \rangle$
- Reachability
Controllable Predecessor

\[ CPre(w) = \tau_1 \land \exists L. \forall N. \delta \rightarrow w' \]
\[ \lor \tau_2 \land \forall L. \forall N. \delta \rightarrow w' \]
Controllable Predecessor

- Iterate to fixed point

![Diagram showing the process of iteratively selecting the predecessor until reaching a fixed point, with nodes representing different states and arrows showing the transition between states. Colors indicate the player's move, with blue for Player 1 and green for Player 2, and the final red node indicating a winning state.]
Controllable Predecessor

- Extract strategy
Symbolic Games

• Variables
  – State: X
  – Label: Y

• $\delta(X, Y, X')$

• $C_{\text{pre}}(\text{win}) = \tau_1 \land \exists l. \forall n. \delta \rightarrow \text{win}'$
  
  $\lor \tau_2 \land \forall l. \forall n. \delta \rightarrow \text{win}'$
Abstraction

- Equivalence relation over states that behave similarly
Abstraction Example

Bool X, Y, P, Q := False;

Action1 : X := True
Action2 : Y := P
Action3 : P := True
Action4 : Q := True

Goal := (X == True) && (Y == True)
Abstraction Example

Goal := (X == True) && (Y == True)

Action1 : X := True
Action2 : Y := Q
Action3 : P := True
Action4 : Q := True

Init

Goal
Three Valued Game Solving

- Abstraction introduces imprecision
- De Alfaro and Roy, 2007
- Classify states
  - Must win
  - May win
  - Must lose
Three Valued Game Solving

\[ CPre(w) = \exists U. \exists L. \forall N. \delta \rightarrow w' \]

Winning?
- Yes if \( Q = T \)
- No if \( Q = F \)
Three Valued Game Solving

- Yes if $Q=T$
- No if $Q=F$

![Diagram showing three valued game solving with init and goal states]
Three Valued Game Solving

![Diagram showing three valued game solving with areas marked as 'Maybe winning' and 'Winning'.]
Three Valued Game Solving

\[ CP_{re}(w) = \forall U. \exists L. \forall N. \delta \rightarrow w' \]
Three Valued Game Solving

Winning?
- Yes if $Q=\text{T}$
- No if $Q=\text{F}$
Three Valued Game Solving

Maybe winning

Winning
Abstraction Refinement

\[ C_{pre\_U}(W) = \exists L. \forall N. \delta \rightarrow W' \]
Abstraction Refinement

Init

Winning

Q

Action3

Init

Winning
Abstraction Refinement

Init

Winning
Abstraction Refinement

![Diagram of Abstraction Refinement]

- **Init**
- **Winning**
Abstraction Refinement

Solve(game, goal) {
    abstraction ← create_initial_abstraction()
}

}
Abstraction Refinement

Solve(game, goal){
    abstraction ← create_initial_abstraction()
    win_must ← False

    while(true) {
        win_must ← solve(win_must);
    }
}
Abstraction Refinement

Solve(game, goal){
    abstraction ← create_initial_abstraction()
    win_must ← False

    while(true) {
        win_must ← solve(win_must);

        if(init → win_must){
            return True;
        } else {
    } else {

    }
}
}
Abstraction Refinement

Solve(game, goal) {
    abstraction ← create_initial_abstraction()
    win_must ← False

    while(true) {
        win_must ← solve(win_must);

        if(init → win_must) {
            return True;
        } else {
            res ← refine_abstraction
            if(!res) {
                return False;
            }
        }
    }
}
Abstraction Refinement

- We did not create an unnecessarily fine abstraction
- We performed refinement on demand
- We reused our previously computed winning set.
- Selection of variable to promote is cheap
- We do not need to recompute entire transition relation on each refinement
Abstraction Refinement

- Performs well if game outcome can be determined from a subset of state
  - Such as unsatisfiable subset of synthesis competition 2014 benchmarks
Predicate Abstraction

write(1, 19)
Predicate Abstraction

write(2, 59)
Predicate Abstraction

write(addr==1, value)

(hours, minutes) -> (hours', minutes)

write(addr==2, value)

(hours, minutes) -> (hours, minutes')
Predicate Abstraction

write(addr==1, value)
(hours, minutes) → (hours', minutes)
write(addr==2, value)
(hours, minutes) → (hours, minutes')

State predicates: (hours = req_hours)
(minutes = req_minutes)
Predicate Abstraction

write(addr==1, value)
(hours, minutes) -> (hours', minutes)

write(addr==2, value)
(hours, minutes) -> (hours, minutes')

State predicates: (hours = req_hours)
(minutes = req_minutes)

Label predicates: (value = req_hours)
(value = req_minutes)
Predicate Abstraction

State predicates: \((\text{hours} = \text{req\_hours})\)
\((\text{minutes} = \text{req\_minutes})\)

Label predicates: \((\text{value} = \text{req\_hours})\)
\((\text{value} = \text{req\_minutes})\)
\((\text{addr} = 1)\)
\((\text{addr} = 2)\)
Predicate Abstraction

HOURS == REQ_HOURS

MINUTES == REQ_MINUTES

addr == 1 && addr == 2
Predicate Abstraction

 addr==1 && addr==2

CLASH!!!
Predicate Abstraction

\[ CPre(w) = \exists L. \forall N. ((\delta \rightarrow w') \land \text{consistent}) \]
Predicate Abstraction

win_may

win_must
Predicate Abstraction

$\text{win\_may}$

$\text{win\_must}$
Predicate Abstraction

win_may

win_must
Predicate Abstraction

Consistency refinements
Three Valued Game Solving

[Diagram showing a 4x4 grid with different colored sections labeled 'Init' and 'Goal'.]

- Init
- Goal
Predicate Abstraction

Consistency refinements
Predicate Abstraction

State space refinements
Implementation

- Part of the Termite driver synthesis toolkit
- GR(1) games
- ~2000 lines of code
- Binary decision diagram based

termite2.org
www.github.com/termite2
# Results

<table>
<thead>
<tr>
<th>Driver</th>
<th>State bits</th>
<th>Abstract state bits</th>
<th>Num refinements</th>
<th>Synthesis time (s)</th>
<th>Lines of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webcam</td>
<td>75908</td>
<td>30</td>
<td>47</td>
<td>462</td>
<td>113</td>
</tr>
<tr>
<td>16450 UART</td>
<td>335</td>
<td>33</td>
<td>60</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>Exynos UART</td>
<td>896</td>
<td>58</td>
<td>54</td>
<td>645</td>
<td>37</td>
</tr>
<tr>
<td>STM SPI</td>
<td>644</td>
<td>31</td>
<td>32</td>
<td>67</td>
<td>24</td>
</tr>
<tr>
<td>Exynos I2C</td>
<td>222</td>
<td>24</td>
<td>21</td>
<td>45</td>
<td>79</td>
</tr>
<tr>
<td>Real time clock</td>
<td>624</td>
<td>25</td>
<td>25</td>
<td>56</td>
<td>84</td>
</tr>
<tr>
<td>IDE disk</td>
<td>952</td>
<td>31</td>
<td>29</td>
<td>285</td>
<td>94</td>
</tr>
</tbody>
</table>

[termite2.org]