Examples of Shape Analysis

We consider program REVERSE which succinctly is represented as:

```
y := null; while x ≠ null do (x ; y ; x : n) := (x : n ; x ; y)
```

We wish to establish that every member of list \( x \) before execution will be a member of list \( y \) after execution.

The initial abstract state after execution of \( y := \text{null} \) is given by:

\[ \text{null} \quad \text{null} \quad \text{null} \]

We wish to establish that every member of list \( x \) before execution will be a member of list \( y \) after execution.

We consider program REVERSE which can succinctly be represented as:

```
(\text{null} ; x \cdot x, w \cdot x) =: (w \cdot x, \text{null} \cdot x, \text{null} \cdot \text{null})
```

Illustration of Analysis
Abstract Computation Continued
It follows that when (and if) \( x = \text{null} \), \( t \) is reachable from \( y \).

Abstract Computation Continued (3)
Consider again the REVERSE program:

\[
y := \text{null};
\]

\[
\text{while } x \neq \text{null} \text{ do } (x; y; x : n) := (x : n; x; y)
\]

We define the predicate \( \text{reach}(u; v) \) which means that there is a chain of next-links leading from the node pointed to by \( u \) to the node to which \( v \) points.

We define the following abstraction mapping:

\[
x - \text{null} = (x = \text{null}) \quad t - \text{null} = (t = \text{null}) \quad \text{reach}(x; t) - \text{null} = \text{true}
\]

Then, we can take as a predicate base the following predicates:

\[
x = \text{null}; t = \text{null}; \quad \text{reach}(x; t) = \text{true} \quad \text{reach}(y; t) = \text{true}
\]

Consider again the REVERSE program:

A Different Approach: Traditional Predicate Abstraction
Examples of Shape Analysis

A. Pnueli

The Abstracted Program

This leads to the following abstract program:

The Abstracted Program
We will proceed to show how the analysis of this pointer manipulating program can be managed automatically within the TLV framework.

 Doing it in TLV

\[
\{ ![x] \text{next} : \forall x \in \{0, 1, \ldots, 4\}, x \neq 0 : (x = \text{null}) \Rightarrow (\exists i, x_i = x) \land (t_i = t \lor t_i = t + 1) \}
\]

for (t = 1; t <= N; t = t + 1)
\begin{verbatim}
MODULE main
DEFINEN:=4;
null:=0;
go:=(x!=null);
Next[0]:=0;
Nextx:=Next[x];
VAR x:0..N;
y:0..N;
t:0..N;
Next:array 1..N of 0..N;
ASSIGN next(x):=case
    go:Nextx;
    1:x;
end;
next(y):=case
    go:x;
    1:y;
end;
next(t):=t;
for(i=1;i<=N;i=i+1)
    {next(Next[i]):=(go & i=x)?y:Next[i];}
\end{verbatim}
Notethatfile reverse4.smv does not specify any initial condition. This is because the initial condition should specify that \( t \) is reachable from \( x \), and this requirement is better specified in the pf file.

File reverse4.mc pf starts with several definition of functions:

```plaintext
Function reach1(x,y) returns an assertion which expresses the property that \( y \) is reachable from \( x \) in 0 or 1 steps.
Function reach2(x,y) returns an assertion which expresses the property that \( y \) is reachable from \( x \) in 2 or less steps.
Function reach3(x,y) returns an assertion which expresses the property that \( y \) is reachable from \( x \) in 3 or less steps.
```

Following these definitions, we set the initial condition of the system to require:

\[
x \text{ null} \leftarrow \text{ null} \wedge \text{ reach}(\text{y},t) \wedge \text{ reach}(x,t).
\]

We then continue to model check that this is an invariant of the system.
File reverse4-mc.pf

funcreach1(x,y);
return x != nil & (next[x] = y) | (x = y);
end--funcreach1(x,y)

funcreach2(x,y);
local r := 0;
for (i in 1...N)
let r := r | reach1(x,i) & reach1(i,y);
end--for(i in 1...N)
return r;
end--funcreach2(x,y)

funcreach4(x,y);
local r := 0;
for (i in 1...N)
let r := r | reach2(x,i) & reach2(i,y);
end--for(i in 1...N)
return r;
end--funcreach4(x,y)

let nil := 0;
print "Model Check Integrity of Terms\n";
let s[1].i := (y = nil) & (t != nil) & reach4(x,t);
call invariance(x = nil -> reach4(y,t));

Program Analysis, NYU, Spring, 2004

Let s[1][t] := t.
let nil := 0.

return x';
end--func reach4(x',t)

end--for (t in 1...N)
let x := (N...1) reach (x',t) & reach2(t,x);
end--for (t in 1...N)
local r := 0;
func reach2(x',t)
end--func reach2(x',t)

end--func reach4(x',t)

end--func reach3(x,t)

end--for (i in 1...N)
local r := 0;
func reach3(x,t)
end--func reach3(x,t)

end--for (i in 1...N)
local r := 0;
func reach2(x,t)
end--func reach2(x,t)

end--for (i in 1...N)
local r := 0;
func reach1(x,t)
end--func reach1(x,t)

end--for (i in 1...N)
In the next step we move to computing and model checking abstraction of Program

\[
\begin{align*}
\text{xnull} & = (x = \text{null}) \\
\text{tnull} & = (t = \text{null}) \\
\text{rx}_t & = \text{reach}(x,t) \\
\text{ry}_t & = \text{reach}(y,t)
\end{align*}
\]
Lecture 10: Examples of Shape Analysis

A. Pnueli

FILE abs-reverse4.smv

MODULE abs-reverse

  FOR ( i = 0 ) ; i < N ; i = i + 1
  NEXT[i] = : !
 }

∀ i. x : T = i
ASSIGN NEXT x = : x

VAR CS : system conc-reverse(N);

NEXT : array 1..N of 0..N;

VAR x : 0..N; t : 0..N;

ASSIGN NEXT[0] = : [0];

for ( i = 1; i <= N; i = i + 1 )
  NEXT[Next[i]] = : (i = x) ? y : Next[i];

MODULE conc-reverse(N)

VAR xnull, tnull, rxt, ryt, xnull, tnull, rxt, ryt, CS : system conc-reverse(N);

VAR xnull, tnull, rxt, ryt, xnull, tnull, rxt, ryt, CS : system conc-reverse(N);

AS : system abs-reverse(N);

VAR CS : system abs-reverse(N);

xt : AS.x
arf : AS.y

File abs-reverse4.smv

Program Analysis, NYU, Spring, 2004
A. Pnueli

This file consists of the following parts:
1. Functions reach1, reach2, and reach=reach4.
2. Abstractions of an assertion, a relation and a Transition System, assuming that the abstraction mapping has been pre-computed.
3. A new procedure check-counter which prints a concretized version of a counter-example.
4. The main part of the program, which defines the abstraction mapping, performs the abstraction of CS into AS and invokes model checking on the abstract system.

This file consists of the following parts:

File abs-reverse4.pf

We will present each part in a separate slide.
Examples of Shape Analysis

A. Pnueli

The Reach Functions

Funcreach1(x,y);
Local result: = \text{CS}.\text{Next}[x] = y \land x \neq \text{nil} \lor (x = y);
Return result;
End--Funcreach1(x,y)

Funcreach2(x,y);
Local r:= 0;
For (i in 1...N)
Let r := r \lor \text{reach1}(x,i) \land \text{reach1}(i,y);
End--For (i in 1...N)
Return r;
End--Funcreach2(x,y)

Funcreach(x,y);
Local r:= 0;
For (i in 1...N)
Let r := r \lor \text{reach2}(x,i) \land \text{reach2}(i,y);
End--For (i in 1...N)
Return r;
End--Funcreach(x,y)
The Abstraction Functions

A. Pnueli

Func abs-assert (phi); Return (phi & abst) for some vars1; End -- abs-assert.

Func concs (st); -- Concretize State "st".

Func abs-trans (rho); Return (rho & abst & abstp) for some all_vars1; End -- Func abs-trans.

To abs-sys;
Let _s[2].i := abs-assert (_s[1].i);
Let _s[2].tn := _s[1].tn;
For (i in 1..._s[1].tn)
Let _s[2].t[i] := abs-trans (_s[1].t[i]);
Let _s[2].jn := _s[1].jn;
For (i in 1..._s[1].jn)
Let _s[2].j[i] := abs-assert (_s[1].j[i]);
End -- For (i in 1..._s[1].tn)
End -- To abs-sys;

End -- Func abs-assert (phi);

End -- Func abs-trans (rho);
End -- Func concs (st);

End -- abs-sys;

Program Analysis, NYU, Spring, 2004
Print Concrete Counter Example

Print "in Concrete Counter-Example follows:"
If (exist (ce[1])) 
Print "n-- State no. " i = n ' ", i = n ' 
End -- For

Print "in I; 1.1.0"
End -- While (j < 0) 
Let j = j - 1 !

Let cc[e] [j] = fast (pred [ce[j]+1], I) & cc[e] [j]' vars 
(0)
Let cc[e] [j] = fast (ce[j], vars) !
Let j = j + 1 !
End -- While (j < c)
Else Break ! End -- If (next)

If (next) Let j = j + 1 ! Let cc[e] [j] = next !
Local next = succj (cc[e] [j], I) & concs (ce[j+1]) !
(0)
Let cc[e] [j] = fast (ce[j], vars) !
Local length (ce) !
Local j = length (ce) !
If (exist (ce[1])) 
Print "in Concrete Counter-Example exists:"
If (exist (ce[1])) 
Print Concrete Counter Example

A. Pnueli

Program Analysis, NYU, Spring, 2004
Main Part of File abs-reverse4.pf

Let vars1 := _s[1].v;
Let vars2 := _s[2].v;
Let all_vars1 := set_union(vars1, prime(vars1));
Let nil := 1;
Let _s[1].i := (y=nil) & (t!=nil) & reach(x,t);
Let abst := (xnull<->(x=nil)) & (tnull<->(t=nil)) & (rxt<->(reach(x,t)) & (ryt<->(reach(y,t)));
Let abstp := prime(abst);

Call Invariance(xnull => ryt, 2);
check_counter;

Program Analysis, NYU, Spring 2004

A. Pnueli

Lecture 10: Examples of Shape Analysis