Verification Condition Generation for Conditional Information Flow

Torben Amtoft Anindya Banerjee

Kansas State University

FMSE, November 2, 2007

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assertions

Object invariants

The Algorithm

mplementation

onclusion

Introduction

Agreement Assertions

bject Invariants

i ne Algorithm

onclusion

Extra Material

► Specify and implement information flow analysis for sequential OO-programs.

- ► Integrate with programmer assertions in style of JML
- Precision:
 - flow-sensitive
 - model also conditional flows

Amtoft & Banerjee

Introduction

Agreement Assertions

Object Invariants

The Algorithm

mplementation

onclusion

Extra Material

 Hoare-like assertions, computing (weakest) preconditions

 Object invariants, cf. Boogie methodology (no expensive alias analysis)

Amtoft & Banerjee

Introduction

Agreement Assertions

Object Invariants

The Algorithm

implementation

onclusion

- ▶ Data is secret (High) or public/observable (Low).
- Confidentiality: High inputs do not influence Low output channels (End-to-end property)
- Typical analyses based on security types, e.g., (int, H), (com, L);
 - ► Flow insensitive [Volpano/Smith/Irvine, Myers,...]
 - Flow sensitive [Hunt/Sands]

Noninterference

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assertions

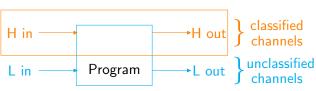
bject Invariants

The Algorithm

Implementation

onclusion

Extra Material



Noninterference property [Goguen-Meseguer]: For any two runs of program, Low-indistinguishable input states yield Low-indistinguishable output states.

Noninterference

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assertions

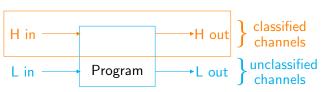
bject Invariants

The Algorithm

Implementation

onclusion

extra Material



Noninterference property [Goguen-Meseguer]: For any two runs of program, Low-indistinguishable input states yield Low-indistinguishable output states.

Equivalently [Cohen]: L out independent of initial H in.

Object Invariants

Conclusion

Extra Material

Amtoft & Banerjee

Consider (Hoare-style) triple [Amtoft/Banerjee SAS'04] $\{x_1 \ltimes \ldots \times x_n \ltimes\} P \{y_1 \ltimes \ldots y_m \ltimes\}$

Meaning: given any two runs of P:

- ▶ If observable inputs $x_1, ..., x_n$ agree (precondition)
- ▶ Then observable outputs $y_1, ..., y_m$ agree in the same two runs (postcondition).

Special case: x_1, \ldots, x_n and y_1, \ldots, y_m are the low variables.

Leveraging Standard Assertions

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

bject Invariants

he Algorithm

Implementation

onclusion

xtra Material

 $\{x \ltimes\}$

if w

then x := 7 else x := 7

Leveraging Standard Assertions

naive rules need w to be low

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Agreement Assertions

 $\{x \ltimes\}$

 $\{w \ltimes\}$

if w

then x := 7else x := 7

no assumptions about w

```
ntroduction
```

Agreement Assertions

bject Invariants

The Algorithm

Implementation

onclusion

```
if w
  then x := 7
  else x := 7
{}
assert(x = 7)
{x \times}
```

```
ntroduction
```

bject Invariants

The Algorithm

Implementation

onclusion

Extra Material

```
else x := 7
\{\}
assert(x = 7)
\{x \bowtie \}
```

if w

then x := 7

Integrate with assertion checker (ESC/Java2, BLAST)

no assumptions about w

Object Invariants

The Algorithm

Implementation

onclusion

extra Material

Recall rule for variable assignment:

$$\{\phi[E/x]\}\ x := E\ \{\phi\}$$

For field update, we could try

$$\{\phi[E/x.f]\} \ x.f := E \{\phi\}$$

so that for example

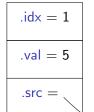
$${w = 7 \land y.f = 5} \ x.f := w \ {x.f = 7 \land y.f = 5}$$

but this is incorrect if x and y alias. (Above is main motivation for separation logic.)

Object Invariants

Extra Material

Motivated by an actual program, provided by Rockwell-Collins, used in hardware verification of operational amplifiers.



X

$$.idx = 2$$

$$.idx = 3$$

$$.val = 8$$

$$.src =$$

Object Invariants

The Algorithm

Implementation

onclusion

Extra Materia

.idx = 3 .val = 8

.src =

y := x.src;

 $.\mathsf{idx} = 1$ $.\mathsf{val} = 5$

.src =

.val = 7

.idx = 2

X

<u>y</u>

Object Invariants

The Algorithm

принентац

Extra Material

y := x.src;

 $t := y.val; \quad x.val := t$

 $.\mathsf{idx} = 1$

.val = 8

.src =

X

.idx = 2

.val = 7

.src =

.idx = 3

.val = 8

.src =

y

Object Invariants

The Algorithm

-

Extra Material

result = 8

y := x.src;t := y.val; x.val := t

result := x.val

.idx = 2

.val = 7

.src =

.idx = 3

.val = 8

.src =

X

.src =

.idx = 1

.val = 8

y

Object Invariants

The Algorithm

...,

xtra Material

Overall policy: odd elements should be public

```
y := x.src; \quad i := x.idx
```

$$t := y.val; \quad x.val := t$$
 $result := x.val$
 $\{odd(i) \Rightarrow result \ltimes\}$

Object Invariants

The Algorithm

.

xtra Material

Overall policy: odd elements should be public

y := x.src; i := x.idx t := y.val; x.val := t result := x.val $\{odd(i) \Rightarrow result \ltimes\}$

Object Flow Invariant (holds for object in steady state)

$$\{ odd(o.idx) \Rightarrow o.val \ltimes \}$$
$$\{ odd(o.idx) \Rightarrow o.src \ltimes \}$$

Object Invariants

The Algorithm

.

Extra Material

Overall policy: odd elements should be public

```
y := x.src; i := x.idx
assert(odd(i) \rightarrow odd(y.idx))
t := y.val; x.val := t
result := x.val
\{odd(i) \Rightarrow result \ltimes\}
```

Object Flow Invariant (holds for object in steady state)

$$\{odd(o.idx) \Rightarrow o.val \ltimes\}$$

 $\{odd(o.idx) \Rightarrow o.src \ltimes\}$

Intuition: to update odd elements, only use odd elements

All objects are manipulated within scopes.Each scope must maintain the object invariant

(cf. pack/unpack in Boogie)

Then aliasing issue become irrelevant.

Amtoft & Banerjee

ntroduction

Agreement Assertions

Object Invariants

The Algorithm

Implementation

nclusion

Extra Material

4 D > 4 P > 4 B > 4 B > B = 999

Object Invariants

Conclusion

```
► All objects are manipulated within scopes.
```

- Each scope must maintain the object invariant (cf. pack/unpack in Boogie)
- ▶ Then aliasing issue become irrelevant.

```
y := x.src; open x \{ y := src; i := .idx \}
```

Object Invariants

Conclusion

```
► All objects are manipulated within scopes.
```

- Each scope must maintain the object invariant (cf. pack/unpack in Boogie)
- ▶ Then aliasing issue become irrelevant.

```
\begin{array}{ll} y := x.src; & \text{open } x \ \{\\ i := x.idx & y := .src; \ i := .idx \ \}\\ \text{assert}(odd(i)) & \text{open } y \ \{\\ \text{assert}(odd(i) \rightarrow odd(.idx));\\ t := y.val; & t := .val \ \} \end{array}
```

Object Invariants

The Algorithm

Implementation

Conclusion

```
► All objects are manipulated within scopes.
```

- Each scope must maintain the object invariant (cf. pack/unpack in Boogie)
- ▶ Then aliasing issue become irrelevant.

```
\begin{array}{ll} y := x.src; & \text{open } x \ \{ \\ i := x.idx \\ \text{assert}(odd(i)) \\ \rightarrow odd(y.idx) \\ t := y.val; \\ x.val := t \\ result := x.val \end{array} \quad \begin{array}{ll} \text{open } x \ \{ \\ y := .src; \ i := .idx \ \} \\ \text{open } y \ \{ \\ \text{assert}(odd(i) \rightarrow odd(.idx)); \\ t := .val \ \} \\ \text{open } x \ \{ \\ val := t; \quad result := .val \ \} \end{array}
```

Object Invariants

7 110 7 11801 111111

...

```
► All objects are manipulated within scopes.
```

- Each scope must maintain the object invariant (cf. pack/unpack in Boogie)
- ▶ Then aliasing issue become irrelevant.

```
\begin{array}{ll} y := x.src; & \text{open } x \ \{ \\ i := x.idx \\ \text{assert}(odd(i)) \\ \rightarrow odd(y.idx) \\ t := y.val; \\ x.val := t \\ result := x.val \end{array} \quad \begin{array}{ll} \text{open } x \ \{ \\ y := .src; \ i := .idx \ \} \\ \text{open } y \ \{ \\ \text{assert}(odd(i) \rightarrow odd(.idx)); \\ t := .val \ \} \\ \text{open } x \ \{ \\ \text{assert}(.idx = i) \\ .val := t; \quad result := .val \ \} \end{array}
```

Amtoft & Banerjee

ntroduction

Agreement Assertions

Object Invariants

The Algorithm

Implementation

onclusion

Extra Material

assert .*idx* = *i*

.val := t

result := .val

close x

open x

Object Invariants

The Algorithm

impiementatio

onclusion

Extra Material

open x

Object invariant:

assert .idx = i

 $odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$

.val := t

result := .val

 $close x \\ odd(i) \Rightarrow result \ltimes$

Object Invariants

The Algorithm

Implementatio

JIICIUSIOII

extra iviateriai

open x

Object invariant:

assert .idx = i

 $odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$

.val := t

 $\begin{array}{l} \textit{result} := .\textit{val} \\ \textit{odd}(\textit{i}) \Rightarrow \textit{result} \ltimes, \; \textit{odd}(.\textit{idx}) \Rightarrow .\textit{val} \ltimes, \; \textit{odd}(.\textit{idx}) \Rightarrow .\textit{src} \ltimes \\ \end{array}$

 $close x \\ odd(i) \Rightarrow result \ltimes$

◆□▶ ◆□▶ ◆■▶ ◆■ ◆○○

Object Invariants

The Algorithm

mpiementatie

JIICIUSIOII

xtra Material

Object invariant:

open x

assert .idx = i

 $odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$

.val := t

$$odd(i) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes result := .val$$

$$odd(i) \Rightarrow result \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes close x$$

$$odd(i) \Rightarrow \textit{result} \ltimes$$

 $odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$

Object invariant:

open x

 $odd(i) \Rightarrow result \times$

Agreement Assertions

Object Invariants

i ne Aigorithm

.

xtra Material

 $\begin{array}{l} \operatorname{assert}. \mathit{idx} = \mathit{i} \\ \mathit{odd}(\mathit{i}) \Rightarrow \mathit{t} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow \mathit{t} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow .\mathit{src} \ltimes \\ .\mathit{val} := \mathit{t} \\ \mathit{odd}(\mathit{i}) \Rightarrow .\mathit{val} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow .\mathit{val} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow .\mathit{src} \ltimes \\ \mathit{result} := .\mathit{val} \\ \mathit{odd}(\mathit{i}) \Rightarrow \mathit{result} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow .\mathit{val} \ltimes, \; \mathit{odd}(.\mathit{idx}) \Rightarrow .\mathit{src} \ltimes \\ \mathit{close} \; \mathit{x} \end{array}$

Object Invariants

he Algorithm

...piemeneacio

ytra Material

Extra Materia

```
Object invariant:
```

$$odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$$

open x

$$odd(i) \wedge .idx = i \Rightarrow t \ltimes, odd(.idx) \wedge .idx = i \Rightarrow .src \ltimes$$
 $assert.idx = i$
 $odd(i) \Rightarrow t \ltimes, odd(.idx) \Rightarrow t \ltimes, odd(.idx) \Rightarrow .src \ltimes$
 $.val := t$
 $odd(i) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$
 $result := .val$
 $odd(i) \Rightarrow result \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$
 $close x$
 $odd(i) \Rightarrow result \ltimes$

Object Invariants

```
Object invariant:
```

 $odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$

 $true \Rightarrow x \ltimes, odd(i) \Rightarrow t \ltimes$ open x $odd(i) \land .idx = i \Rightarrow t \ltimes, odd(.idx) \land .idx = i \Rightarrow .src \ltimes$

assert idx = i $odd(i) \Rightarrow t \ltimes, odd(.idx) \Rightarrow t \ltimes, odd(.idx) \Rightarrow .src \ltimes$

val := t

 $odd(i) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$ result := val

 $odd(i) \Rightarrow result \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$ close x

 $odd(i) \Rightarrow result \times$

Object Invariants

The Algorithm

Implementation

onclusion

Extra Material

```
Object invariant:
```

$$odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$$

$$true \Rightarrow x \ltimes$$

. . .

$$true \Rightarrow x \ltimes, \ odd(i) \Rightarrow t \ltimes$$

open x

$$odd(i) \land .idx = i \Rightarrow t \ltimes, odd(.idx) \land .idx = i \Rightarrow .src \ltimes$$

assert $.idx = i$

$$odd(i) \Rightarrow t \ltimes, odd(.idx) \Rightarrow t \ltimes, odd(.idx) \Rightarrow .src \ltimes$$

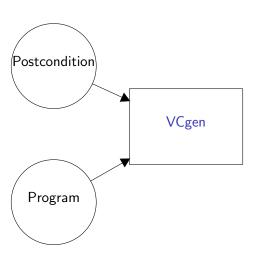
$$.val := t$$

$$odd(i) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$$
 $result := .val$

$$odd(i) \Rightarrow result \ltimes, odd(.idx) \Rightarrow .val \ltimes, odd(.idx) \Rightarrow .src \ltimes$$

$$close x$$

$$odd(i) \Rightarrow result \ltimes$$



Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

bject Invariants

The Algorithm

mplementation

onclusion

Postcondition Precondition **VCgen** Verification Program Conditions

Verification Condition Generation for Conditional Information

Amtoft & Banerjee

ntroduction

..... A ----

bject Invariants

The Algorithm

Implementatio

onclusion

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertio

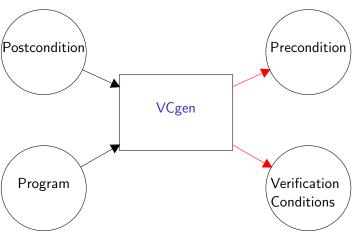
bject Invariants

The Algorithm

mplementation

onclusion

xtra Materia



Always terminates, given loop & object invariants, but VC may fail (if invariants not strong enough)

 θ_1 θ_2 R θ_2' θ_3 Precondition Postcondition **VCgen** Verification Program Conditions

Always terminates, given loop & object invariants, but VC may fail (if invariants not strong enough)

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assert

bject Invariants

The Algorithm

Implementation

onclusion

Extra Mater

Syntax & Semantics

RS ::= skip

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

Object Invariants

The Algorithm

....piementae

Extra Materia

TS ::= skip $\mid assert(\phi)$ $\mid TS ; TS$ $\mid if B then TS TS$ $\mid else TS$ $\mid while B do TS$ $\mid x := A$

new x RSopen x RS

| while B do RS | x := A | .f := A

 $assert(\phi)$

if B then RS

else *RS*

RS : *RS*

Object Invariants

The Algorithm

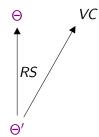
impiementati

Lonclusion

Extra Materia

$$RS ::= \text{skip} \qquad TS ::= \text{skip} \\ \mid \text{assert}(\phi) \qquad \mid \text{assert}(\phi) \\ \mid RS ; RS \qquad \mid TS ; TS \\ \mid \text{if B then RS} \qquad \mid \text{if B then TS TS} \\ \mid \text{else RS} \qquad \mid \text{else TS} \\ \mid \text{while B do RS} \qquad \mid \text{while B do TS} \\ \mid x := A \qquad \mid x := A \\ \mid .f := A \qquad \qquad \mid \text{new x RS} \\ \mid \text{open x RS} \\ \mid \text{s,r $\llbracket RS \rrbracket s',r'} \qquad s,h \, \llbracket TS \rrbracket \, s',h'$$

s: store, r: object state (maps fields to values), h: heap



Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

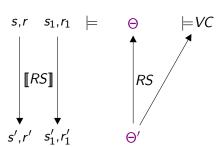
Agreement Assertions

Object Invariants

The Algorithm

Implementation

onclusion



Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assertions

bject Invariants

The Algorithm

mplementation

onclusion

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

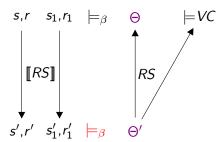
bject Invariants

The Algorithm

mplementation

onclusion

extra Material



► For RS, the heap stays the same

Verification Condition Generation for Conditional Information

Amtoft & Banerjee

Introduction

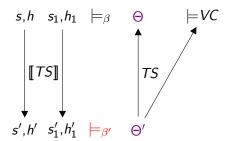
Agreement Assertions

bject Invariants

The Algorithm

mplementation

onclusion



- ► For *RS*, the heap stays the same
- ▶ For TS, the heap may be augmented

Verification Condition Generation for Conditional Information

Amtoft & Banerjee

Introduction

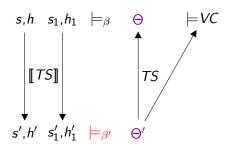
Agreement Assertions

bject Invariants

The Algorithm

mplementation

onclusion



- ► For *RS*, the heap stays the same
- ▶ For *TS*, the heap may be augmented
- ▶ This is termination insensitive

s,h $s_1,h_1 \models_{\beta}$

 $s', h', s'_1, h'_1 \models_{B'}$

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

The Algorithm

▶ For RS, the heap stays the same

- ▶ For TS, the heap may be augmented
- This is termination insensitive
- Auxiliary lemmas tell about R component

TS

write confinement, aka "* property"



Implementation

 $w > 5 \Rightarrow x \times$

Θ

R x := y + z

 Θ' $x > 7 \Rightarrow w \ltimes$

$$[VC]\{\Theta\}\ (R) \longleftarrow x := A\{\Theta'\}$$

 Θ'

ntroduction

Agreement Assertions

bject Invariants

The Algorithm

Implementation

onclusion

Extra Material

$$\Theta \quad y+z > 7 \Rightarrow w \ltimes \qquad w > 5 \Rightarrow (y+z) \ltimes R$$

$$R \quad x := y+z$$

$$[VC]\{\Theta\} (R) \Longleftrightarrow x := A \{\Theta'\}$$
iff $R = \{(\phi[A/x] \Rightarrow E[A/x] \ltimes, \phi \Rightarrow E \ltimes) \mid (\phi \Rightarrow E \ltimes) \in \Theta'$

and
$$\Theta = dom(R)$$
 and $VC = \emptyset$

 $x > 7 \Rightarrow w \times$

 $w > 5 \Rightarrow x \times$

Amtoft & Banerjee

troduction

Agreement Assertions

bject Invariants

The Algorithm

Implementation

onclusion

$$\Theta \quad y + z > 7 \Rightarrow w \ltimes \qquad w > 5 \Rightarrow (y + z) \ltimes$$

$$R \quad \downarrow \qquad \qquad \downarrow \qquad \qquad$$

$$[VC]\{\Theta\}\ (R) \Longleftarrow x := A\ \{\Theta'\}$$
 iff $R = \{(\phi[A/x] \Rightarrow E[A/x] \bowtie, \gamma, \phi \Rightarrow E \bowtie) \mid (\phi \Rightarrow E \bowtie) \in \Theta',$
$$\gamma = m \text{ iff } x \in fv(E)$$
 and $\Theta = dom(R)$ and $VC = \emptyset$

Introduction

Agreement Assertions

object invariant

0

Implementation

nclusion

$$v > 3 \Rightarrow w \bowtie$$

if $y > 5$
 $v > 3 \Rightarrow w \bowtie$
 $v > 3 \Rightarrow w \bowtie$
 $v > 3 \Rightarrow w \bowtie$
 $v > 3 \Rightarrow w \bowtie$

$$\begin{split} [\mathit{VC}]\{\Theta\} \; (R) & \Longleftarrow \text{ if } B \text{ then } S_1 \text{ else } S_2 \; \{\Theta'\} \\ \text{iff} \quad [\mathit{VC}_1]\{\Theta_1\} \; (R_1) & \Longleftarrow S_1 \; \{\Theta'\} \\ \text{and } [\mathit{VC}_2]\{\Theta_2\} \; (R_2) & \Longleftarrow S_2 \; \{\Theta'\} \; \text{and } \mathit{VC} = \mathit{VC}_1 \cup \mathit{VC}_2 \\ \text{and } R & = R_0 \cup R_1' \cup R_2' \cup R_0' \; \text{and } \Theta = \mathit{dom}(R) \\ \text{where } R_0 & = \{(\phi \Rightarrow E \ltimes, u, \theta') \\ & \mid \; (\phi \Rightarrow E \ltimes, u, \theta') \in R_1, \\ & \; (\phi \Rightarrow E \ltimes, u, \theta') \in R_2 \end{split}$$

Amtoft & Banerjee

ntroduction

Agreement Assertions

Object Invariants

The Algorithm

Implementation

onclusion

$$(z > 3 \land y > 5) \lor (v > 3 \land y \le 5) \Rightarrow w \bowtie$$

$$z > 3 \Rightarrow w \bowtie$$

$$x := w$$

$$z > 3 \Rightarrow w \bowtie$$

$$z := v$$

$$\begin{split} [\mathit{VC}]\{\Theta\} \; (R) & \Longleftarrow \text{ if } B \text{ then } S_1 \text{ else } S_2 \; \{\Theta'\} \\ \text{iff} \quad [\mathit{VC}_1]\{\Theta_1\} \; (R_1) & \Longleftarrow S_1 \; \{\Theta'\} \\ \text{and } [\mathit{VC}_2]\{\Theta_2\} \; (R_2) & \Longleftarrow S_2 \; \{\Theta'\} \; \text{and } \mathit{VC} = \mathit{VC}_1 \cup \mathit{VC}_2 \\ \text{and } R & = R_0 \cup R_1' \cup R_2' \cup R_0' \; \text{and } \Theta = \mathit{dom}(R) \\ \text{where } R_0 & = \{(((\phi_1 \land B) \lor (\phi_2 \land \neg B)) \Rightarrow \mathit{E} \bowtie, \mathit{u}, \theta') \\ & \mid \; (\phi_1 \Rightarrow \mathit{E} \bowtie, \mathit{u}, \theta') \in R_1, \\ & (\phi_2 \Rightarrow \mathit{E} \bowtie, \mathit{u}, \theta') \in R_2 \end{split}$$

Implementation

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

 $z > 7 \land v > 5 \Rightarrow w \bowtie$

if v > 5 $z > 7 \Rightarrow w \times$ $v > 7 \Rightarrow x \times$ $z > 7 \Rightarrow x \times$

 $[VC]\{\Theta\}\ (R) \iff \text{if } B \text{ then } S_1 \text{ else } S_2 \{\Theta'\}$ iff $[VC_1]\{\Theta_1\}(R_1) \iff S_1\{\Theta'\}$ and $[VC_2]\{\Theta_2\}$ $(R_2) \longleftarrow S_2$ $\{\Theta'\}$ and $VC = VC_1 \cup VC_2$ and $R = R_0 \cup R_1' \cup R_2' \cup R_0'$ and $\Theta = dom(R)$ where $R'_1 = \{((\phi_1 \land B) \Rightarrow E_1 \ltimes, m, \theta')\}$ $(\phi_1 \Rightarrow E_1 \ltimes, \gamma, \theta') \in R_1$ $\gamma = m \text{ or } (-, m, \theta') \in R_2$

Implementation

Conditional Information Flow

Amtoft & Banerjee

 $z > 7 \land v > 5 \Rightarrow w \bowtie v > 7 \land v < 5 \Rightarrow x \bowtie$

if y > 5 $z > 7 \Rightarrow w \times$ $v > 7 \Rightarrow x \times$ $z > 7 \Rightarrow x \times$

 $[VC]\{\Theta\}\ (R) \iff \text{if } B \text{ then } S_1 \text{ else } S_2 \{\Theta'\}$ iff $[VC_1]\{\Theta_1\}(R_1) \iff S_1\{\Theta'\}$ and $[VC_2]\{\Theta_2\}$ $(R_2) \longleftarrow S_2$ $\{\Theta'\}$ and $VC = VC_1 \cup VC_2$ and $R = R_0 \cup R_1' \cup R_2' \cup R_0'$ and $\Theta = dom(R)$ where $R_2' = \{((\phi_2 \land \neg B) \Rightarrow E_2 \ltimes, m, \theta')\}$ $(\phi_2 \Rightarrow E_2 \ltimes, \gamma, \theta') \in R_2$ $\gamma = m \text{ or } (\underline{\ }, m, \theta') \in R_1$

Introduction

Agreement Assertion

Object Invariants

The Algorithm

Implementation

onclusion

$$z > 7 \land y > 5 \Rightarrow w \ltimes \qquad v > 7 \land y \le 5 \Rightarrow x \ltimes$$

$$(z > 7 \land y > 5) \lor (v > 7 \land y \le 5) \Rightarrow (y > 5) \ltimes$$
if $y > 5$

$$z > 7 \Rightarrow w \ltimes$$

$$x := w$$

$$z > 7 \Rightarrow x \ltimes$$

$$\begin{split} [\mathit{VC}]\{\Theta\} \; (R) &\longleftarrow \text{if } \mathit{B} \text{ then } \mathit{S}_1 \text{ else } \mathit{S}_2 \, \{\Theta'\} \\ &\text{iff} \quad [\mathit{VC}_1]\{\Theta_1\} \; (R_1) \longleftarrow \mathit{S}_1 \, \{\Theta'\} \\ &\text{and } [\mathit{VC}_2]\{\Theta_2\} \; (R_2) \longleftarrow \mathit{S}_2 \, \{\Theta'\} \; \text{and } \mathit{VC} = \mathit{VC}_1 \cup \mathit{VC}_2 \\ &\text{and } \mathit{R} = \mathit{R}_0 \cup \mathit{R}_1' \cup \mathit{R}_2' \cup \mathit{R}_0' \; \text{and } \Theta = \mathit{dom}(\mathit{R}) \\ &\text{where } \mathit{R}_0' = \{(((\phi_1 \land \mathit{B}) \lor (\phi_2 \land \neg \mathit{B})) \Rightarrow \mathit{B} \ltimes, \mathit{m}, \theta') \\ & \mid \; (\phi_1 \Rightarrow \mathit{E}_1 \ltimes, \gamma_1, \theta') \in \mathit{R}_1, \\ & \; (\phi_2 \Rightarrow \mathit{E}_2 \ltimes, \gamma_2, \theta') \in \mathit{R}_2, \\ & \; \gamma_1 = \mathit{m} \; \text{or} \; \gamma_2 = \mathit{m} \end{split}$$

bject Invariants

The Algorithm

Implementation

onclusion

Extra Material

Needs to apply rules like

$$\phi \Rightarrow (x + w) \ltimes \implies \phi \Rightarrow x \ltimes, \ \phi \Rightarrow w \ltimes$$
$$(x = 1 \land x \neq 1) \Rightarrow w \ltimes \implies \mathbf{T} \Rightarrow 0 \ltimes$$
$$x = 1 \Rightarrow x \ltimes \implies \mathbf{T} \Rightarrow 0 \ltimes$$

ntroduction

Agreement Assertions

bject Invariants

The Algorithm

Implementation

Conclusion

Extra Material

▶ implementation (Jonathan Hoag)

- compute loop invariants
- interprocedural (given method summaries)
- array manipulation
- language for conditional information flow (extending SPARK)
- conditional information flow for state chart language

Object invariants

Conclusion

xtra Material

EXTRA MATERIAL

Logic can be augmented by points-to assertions $x \rightsquigarrow L$ [Amtoft/Bandhakavi/Banerjee POPL'06]

- ▶ If also $y \rightsquigarrow L_1$ with L and L_1 disjoint, then x and y cannot alias.
- semantically sound
- sound intraprocedural algorithm

Object Invariants

Conclusion

xtra Material

Logic can be augmented by points-to assertions $x \rightsquigarrow L$ [Amtoft/Bandhakavi/Banerjee POPL'06]

- ▶ If also $y \rightsquigarrow L_1$ with L and L_1 disjoint, then x and y cannot alias.
- semantically sound
- sound intraprocedural algorithm

Problems:

- algorithm needs to be told the shape of the heap (might be overcome by Indus)
- does not easily integrate with programmer assertions
- logic does not capture conditional information flows

Conclusion

Self-composition

 $\{x \ltimes\}$ $\{w \ltimes\}$

is equivalent to (using primes for fresh copies)

$$\begin{cases} x = x' \\ P; \\ P' \\ \{ w = w' \} \end{cases}$$

which can in principle be checked by tool for standard safety analysis.

bject Invariants

Implementation

Conclusion

Extra Material

Self-composition

 $\begin{cases} x \ltimes \} \\ P \\ \{ w \ltimes \} \end{cases}$

is equivalent to (using primes for fresh copies)

$$\begin{cases} x = x' \\ P; \\ P' \\ \{ w = w' \} \end{cases}$$

which can in principle be checked by tool for standard safety analysis.

- ► For good results, need to combine with security static analysis [Terauchi/Aiken, SAS'05]
- ► To deal with heap manipulation, more machinery is needed [Naumann, ESORICS'06]

Amtoft & Banerjee

ntroduction

Agreement Assertions

bject Invariants

....

Implementatio

Conclusion

- ▶ Integrate with automatic safety analysis tool
- compute object invariants
- compute method summaries
- compilation from structured code to state chart, also compiling information flow assertions
- ▶ applications to declassification
- concurrency

Language for Conditional Information Flow

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

bject Invariants

he Algorithm

Implementation

onclusion

Extra Material

SPARK has annotations derives w_1 from x_1, x_2 derives w_2 from x_2, x_3

Language for Conditional Information Flow

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

Object Invariants

The Algorithm

mplementation

onclusion

Extra Material

SPARK has annotations derives w_1 from x_1, x_2 derives w_2 from x_2, x_3



Introduction

Agreement Assertions

bject Invariants

The Algorithm

Implementation

onclusion

```
SPARK has annotations derives w_1 from x_1, x_2 when \phi_1 derives w_2 from x_2, x_3 when \phi_2 \phi_1 \Rightarrow w_1 \ltimes \frac{\psi_1}{R} \Rightarrow x_1 \ltimes \psi_2 \Rightarrow x_2 \ltimes \psi_2 \Rightarrow w_2 \ltimes \frac{\psi_2}{R} \Rightarrow x_3 \ltimes \psi_3 \Rightarrow x_3 \ltimes \psi_4 \Rightarrow x_4 \ltimes \psi_5 \Rightarrow x_4 \ltimes \psi_5 \Rightarrow x_4 \ltimes \psi_5 \Rightarrow x_4 \ltimes \psi_5 \Rightarrow x_5 \Leftrightarrow x_
```

Assertions

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

Introduction

Agreement Assertions

bject Invariants

The Algorithm

onclusion

$$\begin{split} [\mathit{VC}]\{\Theta\} \; (R) & \Longleftrightarrow \mathtt{assert}(\phi_0) \; \{\Theta'\} \\ & \text{iff} \quad R = \{((\phi \land \phi_0) \Rightarrow E \bowtie, u, \phi \Rightarrow E \bowtie) \\ & \quad \mid (\phi \Rightarrow E \bowtie) \in \Theta' \\ & \text{and} \; \Theta = \mathit{dom}(R) \; \text{and} \; \mathit{VC} = \emptyset \end{split}$$

Extra Material

Conditional Information

 $[VC]\{\Theta\}\ (R) \iff \mathtt{assert}(\phi_0)\{\Theta'\}$ iff $R = \{((\phi \land \phi_0) \Rightarrow E \ltimes, u, \phi \Rightarrow E \ltimes)\}$ $|(\phi \Rightarrow E \ltimes) \in \Theta'$ and $\Theta = dom(R)$ and $VC = \emptyset$

$$\Theta$$
 $y < 5 \land y > 7 \Rightarrow z \ltimes$

$$w > 5 \land y > 7 \Rightarrow x \bowtie$$

assert(y > 7)R

 $v < 5 \Rightarrow z \times$

Introduction

Agreement Assertions

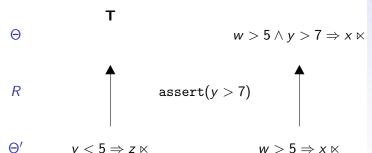
Object Invariants

The Algorithm

Implementation

Extra Material

$$\begin{split} [\mathit{VC}]\{\Theta\} \; (R) &\longleftarrow \mathtt{assert}(\phi_0) \; \{\Theta'\} \\ \text{iff} \quad R &= \{((\phi \land \phi_0) \Rightarrow E \bowtie, u, \phi \Rightarrow E \bowtie) \\ \quad \mid (\phi \Rightarrow E \bowtie) \in \Theta' \\ \text{and} \; \Theta &= \mathit{dom}(R) \; \mathsf{and} \; \mathit{VC} = \emptyset \end{split}$$



ntroduction

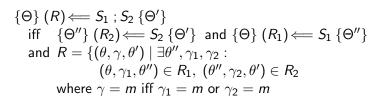
Agreement Assertions

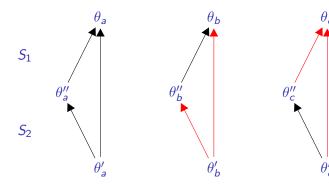
bject Invariants

The Algorithm

Implementation

Conclusion





Introduction

Agreement Assertions

object invariants

. .

onclusion

Extra Material

Procedure p modifies $X = \{x\}$

Precondition for $x \ltimes : \{y > 0 \Rightarrow z \ltimes, y \le 0 \Rightarrow w \ltimes, y \ltimes \}$ derives x from y when y > 0, from y when $y \le 0$

$$\{\Theta\}\ (R) \longleftarrow \mathbf{call}\ p\ \{\Theta'\}$$

iff $R = R_u \cup R_0 \cup R_m$ and $\Theta = dom(R)$

bject Invariants

Implementation

Conclusion

Extra Material

Procedure p modifies $X = \{x\}$

Precondition for $x \ltimes : \{y > 0 \Rightarrow z \ltimes, y \le 0 \Rightarrow w \ltimes, y \ltimes \}$ derives x from y when y < 0, from $y \Leftrightarrow 0$

 $\begin{aligned} \{\Theta\} \; (R) &\longleftarrow \textbf{call} \; p \; \{\Theta'\} \\ &\text{iff} \quad R = R_u \cup R_0 \cup R_m \; \text{and} \; \Theta = \textit{dom}(R) \\ &\text{where} \; R_u = \{(\textit{rm}^+_X(\phi) \Rightarrow E \ltimes, u, \phi \Rightarrow E \ltimes) \\ & \quad \mid (\phi \Rightarrow E \ltimes) \in \Theta' \land \textit{fv}(E) \cap X = \emptyset \} \end{aligned}$

$$y > 0 \Rightarrow w \ltimes$$



$$x > 0 \land y > 0 \Rightarrow w \ltimes$$

bject Invariants

Implementation

Conclusion

Extra Material

Procedure p modifies $X = \{x\}$

Precondition for $x \ltimes : \{y > 0 \Rightarrow z \ltimes, y \le 0 \Rightarrow w \ltimes, y \ltimes \}$ derives x from y when y < 0, from $y \Leftrightarrow 0$

 $\{\Theta\} \ (R) \Longleftarrow \textbf{call} \ p \ \{\Theta'\}$ iff $R = R_u \cup R_0 \cup R_m \text{ and } \Theta = dom(R)$ where $R_0 = \{(rm_X^+(\phi) \Rightarrow v \ltimes, m, \phi \Rightarrow E \ltimes)$ $| \ (\phi \Rightarrow E \ltimes) \in \Theta', \ v \in fv(E) \setminus X \subset fv(E)\}$ $w > 0 \Rightarrow v \ltimes.$

 $y > 0 \Rightarrow w \ltimes$

call p



$$x > 0 \land y > 0 \Rightarrow w \bowtie$$

$$w > 0 \Rightarrow (x + v) \times$$

bject Invariants

The Algorithm

Implementation

Procedure
$$p$$
 modifies $X = \{x\}$
Precondition for $x \ltimes : \{y > 0 \Rightarrow z \ltimes, y \le 0 \Rightarrow w \ltimes, y \ltimes\}$
derives x from y when $y < 0$, from y when $y < 0$

$$\{\Theta\}\ (R) \Longleftarrow \mathbf{call}\ p\ \{\Theta'\}$$
 iff $R = R_u \cup R_0 \cup R_m$ and $\Theta = dom(R)$ where $R_m = \{(rm_X^+(\phi) \land \phi_X \Rightarrow E_X \ltimes, m, \phi \Rightarrow E \ltimes) \mid (\phi \Rightarrow E \ltimes) \in \Theta', \ x \in fv(E) \cap X\}$
$$w > 0 \Rightarrow v \ltimes, \ w > 0 \Rightarrow y \ltimes$$

$$y > 0 \Rightarrow w \ltimes \qquad w > 0 \land y > 0 \Rightarrow z \ltimes$$

$$w > 0 \land y \leq 0 \Rightarrow w \ltimes$$

$$\mathbf{call}\ p$$

$$x > 0 \land y > 0 \Rightarrow w \ltimes \qquad w > 0 \Rightarrow (x + v) \ltimes$$

oject Invariants

The Algorithm

Implementati

onclusion

```
Need iteration
```

```
while i < 7 do
   if odd(i)
   then r := r + v; v := v + h
   else v := x;
   i := i + 1</pre>
```

ject Invariants

Implementation

onclusion

Extra Material

Need iteration

 $\label{eq:while i < 7 do} \begin{subarray}{ll} while i < 7 do \\ if odd(i) \\ then r := r + v; \, v := v + h \\ else \, v := x; \\ i := i + 1 \end{subarray}$

 $r \bowtie$

	\Rightarrow	\bowtie
F		h
F		i
T		r
F		V
F		X

Extra Material

Need iteration

while i < 7 do if odd(i) then r := r + v; v := v + helse v := x; i := i + 1

 $r \bowtie$

		\Rightarrow	\bowtie
F	F		h
F	Т		i
Т	Т		r
F	odd(i)		V
F	F		Χ

ject Invariants

The Algorithm

. . .

Extra Material

Extra Mate

Need iteration

 $r \bowtie$

 $\label{eq:while i < 7 do} \begin{subarray}{ll} while i < 7 do \\ if odd(i) \\ then r := r + v; \, v := v + h \\ else \, v := x; \\ i := i + 1 \end{subarray}$

			\Rightarrow	\bowtie
F	F	F		h
F	Т	T		i
Т	Т	T		r
F	odd(i)	odd(i)		V
F	F	$\neg odd(i)$		X

 $r \bowtie$

Object Invariants

onclusion

Extra Material

Need iteration to reach fixed point

 $\label{eq:while i < 7 do} \begin{subarray}{ll} while i < 7 do \\ if odd(i) \\ then r := r + v; \, v := v + h \\ else \, v := x; \\ i := i + 1 \end{subarray}$

				\Rightarrow	\bowtie
F	F	F	F		h
F	T	T	Т		i
Т	T	T	Т		r
F	odd(i)	odd(i)	odd(i)		V
F	F	$\neg odd(i)$	T		X

While Loops, continued

Verification Condition Generation for Conditional Information Flow

Amtoft & Banerjee

ntroduction

Agreement Assertions

bject Invariants

he Algorithm

Implementation

onclusion

Extra Material

while x > 5 do x := x - 1; y := y + 1

Introduction

Agreement Assertions

bject Invariants

The Algorithm

Implementation

onclusion

while x	>	5	do
x :=	x -	1;	
v :=	v -l	- 1	

	\Rightarrow	×
Т		X
F		у
v > 25		z

Amtoft & Banerjee

Introduction

Agreement Assertions

bject Invariants

The Algorithm

impiementatio

onclusion

while $x > 5$	do
x := x - 1;	
v := v + 1	

		\Rightarrow	\bowtie
Т	Т		X
F	F		y
y > 25	y > 24		Z

Extra Material

Iteration may not terminate.

while x > 5 do x := x - 1;

y := y + 1

			\Rightarrow	\bowtie
Т	Т	Т		X
F	F	F		У
y > 25	y > 24	y > 23		Z

Object Invariants

The Algorithm

mplementation

onclusion

Extra Material

Iteration may not terminate. Use widening.

while
$$x > 5$$
 do $x := x - 1$; $y := y + 1$

				\Rightarrow	\bowtie
Т	Т	Т	Т		X
F	F	F	F		У
<i>y</i> > 25	<i>y</i> > 24	<i>y</i> > 23	Т		Z