Control Flow Analysis and Security

Chiara Bodei[1], ...

[1]Dipartimento di Informatica
Università di Pisa

CALCULI AND MODELS FOR SECURITY
Pisa, September 2007

Outline

Static Analysis: why?

- There are many questions we can ask about a given program.
- Unfortunately, all interesting questions about the behaviour of a program are undecidable, BUT
- we want to solve practical questions

$\Rightarrow$ approximate answers, still precise enough to fuel our applications.
Approximations are *conservative*: all the errors lean to the same side

Efficiency Concerns

- Transition systems: usually huge $\Rightarrow$ their exploration can be computationally hard.
- Need of obtaining information about the dynamic behaviour, without spending so much.
- There are two alternatives:

... two alternatives:

- Looking through the glass: magic techniques.

- Looking at the system description: static analysis techniques

Static Analysis Techniques

Static techniques {
- Abstract Interpretation
- Control Flow Analysis (CFA)
- Type Systems

- Non trivial information about the dynamic behaviour, by simply inspecting the description of the system.
- predict safe and computable approximations of dynamic behaviuor
- analyse properties that hold in every execution
- give a repertoire of automatic and decidable methods and tools

In Checking properties

STATICALLY (analyse the TEXT)	DYNAMICALLY (analyse the BEHAVIOUR)
approximate	precise
terminates	may *not* terminate
“low” complexity	“high” complexity
“cheap” tools	“expensive” tools

SOUNDNESS

P has the static property implies *P* has the dynamic property

err on the safe side

Abstract Interpretation

- It approximates the concrete semantics of dynamic systems, by giving a corresponding abstract semantics.
- It treats the semantic correctness.
- It may use the abstract semantics as a basis for producing automatic tools.

$$\boxed{+3,\ +5,\ +16} \xrightarrow{\alpha} \boxed{+} \xrightarrow{\gamma} \boxed{+\ \boxed{+3,\ +5,\ +16}}$$

Type Systems

- It divides program values into types and establishes a type discipline.
- It checks whether violations to the type discipline may arise (Type checking). Violations correspond to illegal program behaviour.
- It treats the semantic correctness.

It collects data and, at the same time, it checks them to prove properties.
If 4 : *integer* then **if** 4 **then skip** gives an error.

Control Flow Analysis (CFA)

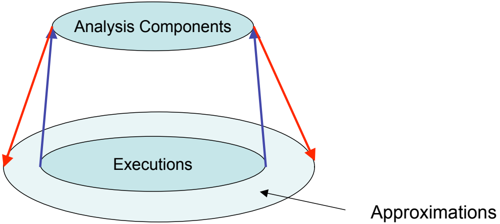
Properties

- It predicts approximations (estimates) to the set of values that the objects of a program may assume at run-time.
- It treats (i) semantic correctness, (ii) existence of least estimates and (iii) efficient construction of least estimates.

1. The analysis collects data and afterwards,
2. it checks them in order to prove properties.

The Nature of Approximation

- CFA represents an abstraction of the actual executions
- Concretisation cannot be precise.



Static	Dynamic
Event E is included	E can happen
Event E is not included	E never happens

From TEXT to ABSTRACT BEHAVIOUR

How is abstract behaviour extracted from program text?

and how is it related to the dynamic behaviour, i.e. soundness?

Programs are annotated (by the compiler) on

- objects (data, vars,. . .)
  (cf. types: x : *real*; f : $real \times nat \to nat$)
- control points (calling points, declaration/use, . . .)

CFA pattern

- Choose those values of interest for the language.
  - Define the shape of estimates.
  - Define a number of clauses.
- Prove that all estimates are semantically correct.
- Prove that least estimate exist.
- Derive a *constructive* procedure that builds estimates.
- Select a specific dynamic security property and define a static check on estimates.

CFA vs Type Systems

- Type Systems are prescriptive, i.e. they infer types and impose the well-formedness conditions at the same time.
- Control Flow Analysis is descriptive, i.e. it merely infers the information and then leaves it to a separate step to actually impose demands on when programs are well-formed.
- For each property, it is often the case that:
  - a new *ad hoc* type system is necessary, while
  - only a new test on the *same* CFA analysis is needed.

Why (S)pi-calculus?

Pi-Calculus

- Communication primitives: simple and powerful.
- Scoping Rules: explicitly control the access to channels and to data.

To know the **name** of a channel amounts to having the capability to communicate on it.
The Pi-Calculus does not include cryptographic primitives.

Spi-Calculus

- Primitives for encryption and decryption.
- Directly executable
- Formal semantics.

π-calculus: Remind

Processes:

$$P ::= \mathbf{0} \mid \mu.P \mid P \mid P \mid (\nu x)P \mid [x{=}y]P \mid\ !P \mid P + P$$

where: $\mu ::= x(y) \mid \overline{x}y \mid \tau$

Communication:

$$\frac{P \xrightarrow{x(y)} P',\ Q \xrightarrow{\overline{x}a} Q'}{P|Q \xrightarrow{\tau} P'\{a/y\}|Q'}$$

ABSTRACT BEHAVIOUR

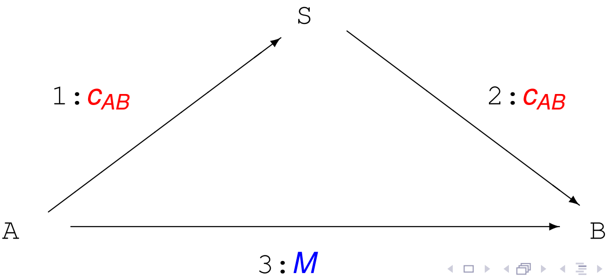
How is abstract behaviour described? As a triple of functions

$$(\rho, \kappa)$$

ρ : name $\mapsto$ {set of names} it can be bound to
κ : binder/name $\mapsto$ {set of names} that can be sent over it.

CFA Example: Wide Mouthed Frog

$$
\begin{array}{lcl}
P & = & (\nu c_{AS})(\nu c_{BS})(\ (A|B)\ |S) \\
A & = & (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S & = & c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B & = & c_{BS}(w).w(y)
\end{array}
$$



CFA Example: Wide Mouthed Frog (2)

$$
\begin{array}{rcl}
P & = & (\nu c_{AS})(\nu c_{BS})(\ (A|B)\ |S) \\
A & = & (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S & = & c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B & = & c_{BS}(w).w(y)
\end{array}
$$

$$
\begin{array}{rclcrcl}
\rho(x) & \supseteq & \{c_{AB}\} & \quad & \kappa(c_{AS}) & \supseteq & \{c_{AB}\} \\
\rho(w) & \supseteq & \{c_{AB}\} & & \kappa(c_{BS}) & \supseteq & \{c_{AB}\} \\
\rho(y) & \supseteq & \{M\} & & \kappa(c_{AB}) & \supseteq & \{M\}
\end{array}
$$

Estimates Validation

When is (ρ, κ) a *valid* estimate? When it satisfies a set of logical clauses. Zoom on three:

$$(\rho, \kappa) \models P \mid Q \text{ iff } (\rho, \kappa) \models P \wedge (\rho, \kappa) \models Q$$

$$(\rho, \kappa) \models \overline{x}\langle y \rangle.P \textbf{ iff } (\rho, \kappa) \models P \wedge \boxed{\forall a \in \rho(x) : \rho(y) \subseteq \kappa(a)}$$

$$(\rho, \kappa) \models x(y).P \text{ iff} (\rho, \kappa) \models P \wedge \boxed{\forall a \in \rho(x), \kappa(a) \subseteq \rho(y)}$$

CFA Properties

SOUNDNESS If $(\rho, \kappa) \models P$ and $P \rightarrow Q$ then $(\rho, \kappa) \models Q$
(sometimes additional assumptions are needed)

EXISTENCE The set $\{(\rho, \kappa) \mid (\rho, \kappa) \models P\}$ is a Moore family*
Therefore there always exists a least solution (ρ, κ).

CONSTRUCTION There is a constructive procedure for obtaining the least solution.

(*) A set $\mathcal{I}$ of proposed estimates is a *Moore family* if and only if it contains $\sqcap \mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}$.

CFA pattern (2)

- Choose those values of interest for the language and define estimates and clauses.
- Prove that all estimates are semantically correct.
- Prove that least estimate exist.
- Derive a *constructive* procedure that builds estimates.
- Select a specific dynamic security property and define a static check on estimates.

This may require to refine estimates.

CFA pattern (2)

- Choose those values of interest for the language and define estimates and clauses.
- Prove that all estimates are semantically correct.
- Prove that least estimate exist.
- Derive a *constructive* procedure that builds estimates.
- Select a specific dynamic security property and define a static check on estimates.

This may require to refine estimates.

Now comes Security: a simple **Secrecy** property

Dynamic notion: Carefulness

No secret datum flows on a public channel.

Static notion: Confinement

1. partition names in $\begin{cases} \mathcal{S} & \text{Secret names} \\ \mathcal{P} & \text{Public names} \end{cases}$
2. compute $(\rho, \kappa) \models P$
3. check that $\kappa(a \in \mathcal{P}) \subseteq \mathcal{P}$

Example

$$
\begin{aligned}
P &= (\nu c_{AS})(\nu c_{BS})(\ (A|B)\ |S) \\
A &= (\nu c_{AB})(\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle) \\
S &= c_{AS}(x).\overline{c_{BS}}\langle x\rangle \\
B &= c_{BS}(w).w(y)
\end{aligned}
$$

$$
\begin{array}{lllllll}
\rho(x) & \supseteq & \{c_{AB}\} & \quad & \kappa(c_{AS}) & \supseteq & \{c_{AB}\} \\
\rho(w) & \supseteq & \{c_{AB}\} & & \kappa(c_{BS}) & \supseteq & \{c_{AB}\} \\
\rho(y) & \supseteq & \{M\} & & \kappa(c_{AB}) & \supseteq & \{M\} \quad \leftarrow
\end{array}
$$

If $\mathcal{S} = \{M, c_{AS}, c_{BS}\}$ and $\mathcal{P} = \{c_{AB}\}$, then P has leaks.

$$\mathcal{S} \ni M \in \kappa(c_{AB} \in \mathcal{P}) \not\subseteq \mathcal{P}$$

The process would have no leaks if c_{AB} and M, were secret or public, respectively.

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance
NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^{l} \mid (\nu x)S \mid S|S \mid !S$,with l level label;
- Semantics: $\dfrac{P \xrightarrow{\mu} Q}{\langle P \rangle^{l} \xrightarrow{\mu, l} \langle Q \rangle^{l}}$
- Analysis: $(\rho, \kappa, \sigma) \models_{l} P$, with $\sigma = \langle \sigma_{in}, \sigma_{out} \rangle$, where
  - $\sigma_{in}(l)$: set of the channels that can be received by an input within a sub-process with level l.
  - $\sigma_{out}(l)$: set of the channels that can be sent by an output within a sub-process with level l.

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance
NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^l \mid (\nu x)S \mid S|S \mid !S$,with l level label;
- Semantics: $$\frac{P \xrightarrow{\mu} Q}{\langle P \rangle^l \xrightarrow{\mu, l} \langle Q \rangle^l}$$
- Analysis: $(\rho, \kappa, \sigma) \models_l P$, with $\sigma = \langle \sigma_{in}, \sigma_{out} \rangle$, where
  - $\sigma_{in}(l)$: set of the channels that can be received by an input within a sub-process with level l.
  - $\sigma_{out}(l)$: set of the channels that can be sent by an output within a sub-process with level l.

Another Property: No Read Up/No Write Down (NRU/NWD)

Processes are given levels of security clearance
NRU/NWD: the sender has a clearance level lower than the level of the receiver.

- Syntax: $S ::= \langle P \rangle^l \mid (\nu x)S \mid S|S \mid !S$,with l level label;
- Semantics: $$\frac{P \xrightarrow{\mu} Q}{\langle P \rangle^l \xrightarrow{\mu, l} \langle Q \rangle^l}$$
- Analysis: $(\rho, \kappa, \sigma) \models_l P$, with $\sigma = \langle \sigma_{in}, \sigma_{out} \rangle$, where
  - $\sigma_{in}(l)$: set of the channels that can be received by an input within a sub-process with level l.
  - $\sigma_{out}(l)$: set of the channels that can be sent by an output within a sub-process with level l.

NRU/NWD: CFA rules

- $(\rho, \kappa, \sigma) \models^{l} \overline{x}y.P$ iff $(\rho, \kappa, \sigma) \models^{l} P \wedge \forall a \in \rho(x) : \left(\begin{array}{l} \rho(y) \subseteq \kappa(a) \wedge \\ \rho(y) \subseteq \sigma_{out}(l)(a)) \end{array} \right)$
- $(\rho, \kappa, \sigma) \models^{l} x(y).P$ iff $(\rho, \kappa, \sigma) \models^{l} P \wedge \forall a \in \rho(x) : \left(\begin{array}{l} \kappa(a) \subseteq \rho(y) \wedge \\ \kappa(a) \subseteq \sigma_{in}(l)(x) \end{array} \right)$
- $(\rho, \kappa, \sigma) \models \langle P \rangle^{l}$ iff $(\rho, \kappa, \sigma) \models^{l} P$

NRU/NWD

Dynamic Property: **no read-up/no write-down**

A high level process cannot write any value to a process at low level; symmetrically a process at low level cannot read data from one of a high level.

Static Property: **discreet**

Each channel cannot be used for sending an object from a process with high level l to a process with low level l'.

$$\forall l', l \text{ with } l' \text{ below } l : \forall \chi : \sigma_{out}(l)(x) \cap \sigma_{in}(l')(x) = \emptyset.$$

SPI

T	$::=$	**Terms**
	...	...
	$\{M_1, \ldots, M_k\}_N$	*shared – key encryption* $(k \geq 0)$

P	$::=$	**Processes**
	...	...
	decrypt L *as* $\{x_1, \ldots, x_k\}_N$ *in* P	*shared – key decryption* $(k \geq 0)$

SPI (2)

SPI extends π with:

- numbers and terms (pairs, encryption)
  typical term M: $\{M_1, \ldots, M_n\}_K$
- decryption primitive
  $\texttt{decrypt}\ \{M_1, \ldots, M_n\}_K\ \texttt{as}\ \{y_1, \ldots, y_n\}_K\ \texttt{in}\ P$
  becomes $P[M_1/y_1, ..., M_n/y_n]$

In $\texttt{decrypt}\ L\ \texttt{as}\ \{x\}'_K\ \texttt{in}\ P$
the process attempts to decrypt L with the key K'; if $L = \{M\}'_K$,
then the process behaves as $P[M/x]$; otherwise it stucks.

How to obtain the CFA for SPI

There are new values of interest: those for terms. $\Rightarrow$ The analysis should track the way terms are manipulated.

- Another extension to the syntax: "labels" assigned to the occurrences of terms.
- A new component of estimates: ζ that associates abstract values with each datum and its components.
- estimates then have the form (ρ, κ, ζ).
- The other ingredients of the analysis are quite similar to the one developed for the π-calculus.

SPI: Estimates

Terms carry labels: M^{l}

An estimate is a triple (ρ, κ, ζ)
$\rho(x)$ — values bound to x
$\kappa(a)$ — values flowing on a
$\zeta(l)$ — values that M (at l) can assume

$$(\rho, \kappa, \zeta) \models \{M_1^{l_1}, \cdots, M_n^{l_n}\}^{l}_{M_0^{l_0}} \text{ iff } \quad \forall i : (\rho, \kappa, \zeta) \models M_i^{l_i} \wedge$$

$$\boxed{\{\zeta(l_1), \cdots, \zeta(l_n)\}_{\zeta(l_0)} \subseteq \zeta(l)}$$

SPI: Secrecy

1. Extend
   - the dynamic notion of CAREFULNESS
   - the static notion of CONFINEMENT
2. Prove that P confined implies P careful
3. Prove that P is careful iff P guarantees Dolev-Yao secrecy.

(For every attacker E, P never sends a secret message that can be intercepted and acquired by E)

Confinement for SPI

Names are partitioned into $\mathcal{S}$ (Secret) and $\mathcal{P}$ (Public).

Values are partitioned as well, depending on their components.

$$kind(w) = \begin{cases} \mathcal{S} & \text{if one of the component of w is } \mathcal{S} \\ \mathcal{P} & \text{otherwise} \end{cases}$$

Static Property for SPI: Confinement

Only public values can be transmitted along a public channel.
For all a in P: $kind(a) = \mathcal{P}$

The Nature of Imprecision

This Control Flow Analysis is Context-Insensitive and is called 0-CFA.

$$P_1 = (a(y) |\ \overline{a}b) \qquad P_2 = (a(y) + \overline{a}b) \qquad P_3 = (a(y).\overline{a}b)$$

- Note that the variable y in P_1 can be bound to b, while in P_2 and in P_3 it cannot.
- Instead, in the CFA estimate, we have that $\rho(y) \supseteq \{b\}$ in all the three cases.