

# JAIL: Javacard Abstract Interpretation-based Lifeguard

Pietro Ferrara

Università Ca' Foscari di Venezia  
Dipartimento di Informatica  
Ecole Polytechnique, Paris

Pisa, 16 febbraio 2006

# Obiettivi

Abbiamo implementato un analizzatore statico generico per Java Card parametrizzato:

- sulla proprietà da analizzare
- sul dominio astratto usato per catturare informazioni sulle variabili numeriche
- su alcuni valori numerici per permettere all'utente di decidere se avere un'analisi più lenta e precisa o più veloce e approssimata

# Java Card

- 1 Sottoinsieme di Java per lo sviluppo e l'implementazione di applicazioni per smart card
- 2 Limitate risorse hardware a disposizione, supporta solo un subset di Java (non supporta il multithreading)
- 3 Essenziali le proprietà di sicurezza, in particolare è definita la proprietà di applet isolation dal JavaCard Runtime Environment

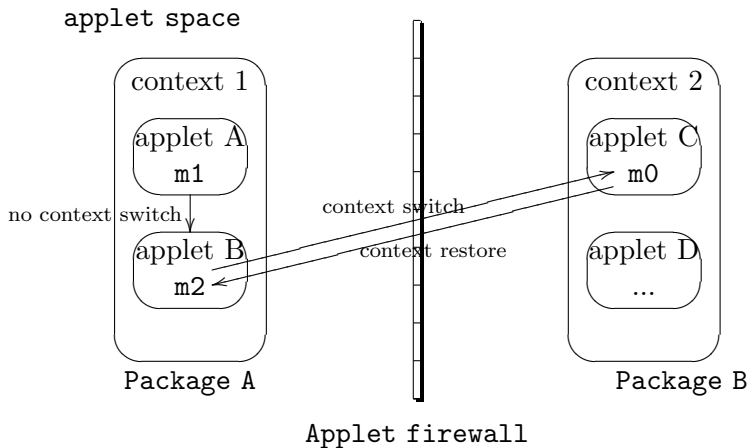


Figure: Context switching

# Semantica Concreta

Vogliamo avere un dominio semantico che supporti l'aliasing di oggetti e che permetta a ciascun oggetto di avere il proprio ambiente:

- Ambiente: funzione parziale che, dato il nome di una variabile, ritorna il suo indirizzo ( $E : N \rightarrow A$ );
- Store: funzione parziale che, dato un indirizzo, ritorna il valore contenuto in esso ( $S : A \rightarrow V$ );
- Valore: può essere un oggetto (il cui stato è contenuto in un ambiente), un valore numerico o un array ( $V = E \cup PV \cup AR$ );
- Funzione delle eccezioni: funzione che, dato il nome dell'eccezione, ritorna lo stato in cui è stata lanciata l'eccezione ( $EX : N \rightarrow ST$ );
- Stato: tripla composta da un ambiente, uno store e una funzione delle eccezioni ( $ST = E \times S \times EX$ )

# Semantica Astratta - Indirizzi

- nel concreto un indirizzo essenzialmente è un valore numerico
- nell'astratto non possiamo usare un contatore numerico che viene incrementato ogni volta che si crea una nuova cella nello store. In questo modo infatti non è garantita la convergenza dell'analisi
- $A^\# = P \times Int \times Stack$  dove
  - il primo valore intero identifica univocamente il punto del codice sorgente che ha allocato l'indirizzo ( $P$  è l'insieme degli identificatori univoci dei punti del programma)
  - il secondo valore contiene il numero di iterazioni effettuati nell'analisi di un ciclo while, nel caso in cui l'indirizzo venga allocato all'interno di un tale ciclo
  - lo stack contiene le chiamate a metodi effettuate fino ad arrivare al punto in cui la cella viene allocata

# Semantica Astratta - Ambienti

A differenza della semantica concreta, nell'astratto una variabile può essere collegata a uno o più indirizzi:  $E^\# : N \rightarrow \wp(A^\#)$

- Relazione d'ordinamento:

$$\hat{e}_1 \leq_{E^\#} \hat{e}_2 \iff \begin{aligned} & \text{dom}(\hat{e}_1) \subseteq \text{dom}(\hat{e}_2) \wedge \\ & \forall n \in \text{dom}(\hat{e}_1) : \hat{e}_1(n) \subseteq \hat{e}_2(n) \end{aligned}$$

- Funzione d'astrazione:

$$\begin{aligned} \alpha'_E(e) = \hat{e} : N \rightarrow \wp(A)^\# \text{ such that} \\ \text{dom}(\hat{e}) = \text{dom}(e) \wedge \forall n \in \text{dom}(e) : \hat{e}(n) = \alpha'_A(e(n)) \end{aligned}$$

- Funzione di concretizzazione:

$$\begin{aligned} \gamma_E(\hat{e}) = \{e_i : N \rightarrow A \mid i \in [1..n]\} \text{ such that} \\ \forall i \in [1..n] : \text{dom}(e_i) = \text{dom}(\hat{e}) \wedge \forall n \in \text{dom}(\hat{e}) : e_i(n) \in \gamma_A(\hat{e}(n)) \\ \wedge \forall n \in \text{dom}(\hat{e}) : \bigcup_{i \in [1..n]} e_i(n) = \gamma_A(\hat{e}(n)) \end{aligned}$$

# Semantica Astratta - Store

$$S^\# : A^\# \rightarrow V^\#$$

- Relazione d'ordinamento:

$$\hat{s}_1 \leq_{S^\#} \hat{s}_2 \iff \text{dom}(\hat{s}_1) \subseteq \text{dom}(\hat{s}_2) \\ \forall \hat{a} \in \text{dom}(\hat{s}_1) : \hat{s}_1(\hat{a}) \leq_{V^\#} \hat{s}_2(\hat{a})$$

- Funzione d'astrazione:

$$\alpha'_S(s) = \hat{s} : A^\# \rightarrow V^\# \text{ such that} \\ \text{dom}(\hat{s}) = \bigcup_{a \in \text{dom}(s)} \alpha'_A(a) \\ \wedge \forall a \in \text{dom}(s) : \hat{s}(\alpha'_A(a)) = \alpha'_V(s(a))$$

- Funzione di concretizzazione:

$$\gamma_S(\hat{s}) = \{s_i : A \rightarrow V \mid i \in [1..n]\} \text{ such that} \\ \forall i \in [1..n] : \text{dom}(s_i) = \bigcup_{\hat{a} \in \text{dom}(\hat{s})} \gamma_A(\hat{a}) \\ \wedge \forall \hat{a} \in \text{dom}(\hat{s}), a \in \gamma_A(\hat{a}) : s_i(a) \in \gamma_V(\hat{s}(\hat{a})) \\ \wedge \forall \hat{a} \in \text{dom}(\hat{s}) : \bigcup_{i \in [1..n], a \in \gamma_A(\hat{a})} s_i(a) = \gamma_V(\hat{s}(\hat{a}))$$

# Semantica Astratta - Funzione delle eccezioni

$EX^\# : N \rightarrow ST^\#$

- Relazione d'ordinamento:

$$\begin{aligned} \hat{ex}_1 \leq_{EX^\#} \hat{ex}_2 &\iff \text{dom}(\hat{ex}_1) \subseteq \text{dom}(\hat{ex}_2) \\ &\wedge \forall n \in \text{dom}(\hat{ex}_1) : \hat{ex}_1(n) \leq_{ST^\#} \hat{ex}_2(n) \end{aligned}$$

- Funzione d'astrazione:

$$\begin{aligned} \alpha'_{EX}(ex) = \hat{ex} : N \rightarrow ST^\# \text{ such that} \\ \text{dom}(\hat{ex}) = \text{dom}(ex) \wedge \forall n \in \text{dom}(ex) : \hat{ex}(n) = \alpha'_{ST}(ex(n)) \end{aligned}$$

- Funzione di concretizzazione:

$$\begin{aligned} \gamma_{EX}(\hat{ex}) = \{ex_i : N \rightarrow ST \mid i \in [1..n]\} \text{ such that} \\ \forall i \in [1..n] : \text{dom}(ex_i) = \text{dom}(\hat{ex}) \\ \wedge \forall n \in \text{dom}(\hat{ex}) : ex_i(n) \in \gamma_{ST}(\hat{ex}(n)) \\ \wedge \forall n \in \text{dom}(\hat{ex}) : \bigcup_{i \in [1..n]} ex_i(n) = \gamma_{ST}(\hat{ex}(n)) \end{aligned}$$

# Semantica Astratta - Stati

$$ST^\# = E^\# \times S^\# \times EX^\#$$

- Relazione d'ordinamento:

$$\hat{st}_1 \leq_{ST^\#} \hat{st}_2 \iff \hat{e}_1 \leq_{E^\#} \hat{e}_2 \wedge \hat{s}_1 \leq_{S^\#} \hat{s}_2 \wedge \hat{ex}_1 \leq_{EX^\#} \hat{ex}_2$$

- Funzione d'astrazione:

$$\alpha'_{ST}((e, s, ex)) = (\alpha'_E(e), \alpha'_S(s), \alpha'_{EX}(ex))$$

- Funzione di concretizzazione:

$$\begin{aligned} \gamma_{ST}((\hat{e}, \hat{s}, \hat{ex})) &= (e, s, ex) \text{ such that} \\ &e \in \gamma_E(\hat{e}), s \in \gamma_S(\hat{s}), ex \in \gamma_{EX}(\hat{ex}) \wedge \\ &\bigcup_{st=(e,s,ex) \in \gamma_{ST}((\hat{e}, \hat{s}, \hat{ex}))} e = \gamma_E(\hat{e}), \\ &\bigcup_{st=(e,s,ex) \in \gamma_{ST}((\hat{e}, \hat{s}, \hat{ex}))} s = \gamma_S(\hat{s}), \\ &\bigcup_{st=(e,s,ex) \in \gamma_{ST}((\hat{e}, \hat{s}, \hat{ex}))} ex = \gamma_{EX}(\hat{ex}) \end{aligned}$$

# Operatore di least upper bound (1/2)

- Ambienti:

$$\hat{e}_1 \sqcup_{E\#} \hat{e}_2 = \{ \hat{e} : n \mapsto \hat{a} \mid \hat{a} = \begin{cases} \hat{e}_1(n) & \text{if } n \in \text{dom}(\hat{e}_1) \setminus \text{dom}(\hat{e}_2) \\ \hat{e}_2(n) & \text{if } n \in \text{dom}(\hat{e}_2) \setminus \text{dom}(\hat{e}_1) \\ \hat{e}_1(n) \cup \hat{e}_2(n) & \text{if } n \in \text{dom}(\hat{e}_1) \cap \text{dom}(\hat{e}_2) \end{cases} \}$$

- Store:

$$\hat{s}_1 \sqcup_{S\#} \hat{s}_2 = \{ \hat{s} : \hat{a} \mapsto \hat{v} \mid \hat{v} = \begin{cases} \hat{s}_1(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_1) \setminus \text{dom}(\hat{s}_2) \\ \hat{s}_2(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_2) \setminus \text{dom}(\hat{s}_1) \\ \hat{s}_1(\hat{a}) \sqcup_{V\#} \hat{s}_2(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_1) \cap \text{dom}(\hat{s}_2) \end{cases} \}$$

# Operatore di least upper bound (2/2)

- Eccezioni:

$$\hat{e}x_1 \sqcup_{EX\#} \hat{e}x_2 = \{ \hat{e}x : n \mapsto \hat{st} \mid \hat{st} = \begin{cases} \hat{e}x_1(n) & \text{if } n \in \text{dom}(\hat{e}x_1) \setminus \text{dom}(\hat{e}x_2) \\ \hat{e}x_2(n) & \text{if } n \in \text{dom}(\hat{e}x_2) \setminus \text{dom}(\hat{e}x_1) \\ \hat{e}x_1(n) \sqcup_{ST\#} \hat{e}x_2(n) & \text{if } n \in \text{dom}(\hat{e}x_1) \cap \text{dom}(\hat{e}x_2) \end{cases} \}$$

- Stati:

$$\hat{st}_1 \sqcup_{ST\#} \hat{st}_2 = (\hat{e}_1 \sqcup_{E\#} \hat{e}_2, \hat{s}_1 \sqcup_{S\#} \hat{s}_2, \hat{e}x_1 \sqcup_{EX\#} \hat{e}x_2)$$

# Operatore di greatest lower bound

- Ambienti:

$$\hat{e}_1 \sqcap_{E\#} \hat{e}_2 = \{\hat{e} : n \mapsto \hat{a} \mid \hat{a} = \hat{e}_1(n) \cap \hat{e}_2(n) \text{ if } n \in \text{dom}(\hat{e}_1) \cap \text{dom}(\hat{e}_2)\}$$

- Store:

$$\hat{s}_1 \sqcap_{S\#} \hat{s}_2 = \{\hat{s} : \hat{a} \mapsto \hat{v} \mid \hat{v} = \hat{s}_1(\hat{a}) \cap_{V\#} \hat{s}_2(\hat{a}) \text{ if } \hat{a} \in \text{dom}(\hat{s}_1) \cap \text{dom}(\hat{s}_2)\}$$

- Eccezioni:

$$\hat{e}x_1 \sqcap_{EX\#} \hat{e}x_2 = \{\hat{e}x : n \mapsto \hat{st} \mid \hat{st} = \hat{e}x_1(n) \cap_{ST\#} \hat{e}x_2(n) \\ \text{if } n \in \text{dom}(\hat{e}x_1) \cap \text{dom}(\hat{e}x_2)\}$$

- Stati:

$$\hat{st}_1 \sqcap_{ST\#} \hat{st}_2 = (\hat{e}_1 \sqcap_{E\#} \hat{e}_2, \hat{s}_1 \sqcap_{S\#} \hat{s}_2, \hat{e}x_1 \sqcap_{EX\#} \hat{e}x_2)$$

# Operatore di widening

- Ambienti:  $\hat{e}_1 \nabla_{E\#} \hat{e}_2 = \hat{e}_1 \sqcup_{E\#} \hat{e}_2$
- Store:

$$\hat{s}_1 \nabla_{S\#} \hat{s}_2 = \{ \hat{s} : \hat{a} \mapsto \hat{v} \mid \hat{v} = \begin{cases} \hat{s}_1(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_1) \setminus \text{dom}(\hat{s}_2) \\ \hat{s}_2(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_2) \setminus \text{dom}(\hat{s}_1) \\ \hat{s}_1(\hat{a}) \nabla_{V\#} \hat{s}_2(\hat{a}) & \text{if } \hat{a} \in \text{dom}(\hat{s}_1) \cap \text{dom}(\hat{s}_2) \end{cases} \}$$

- Eccezioni:

$$\hat{e}x_1 \nabla_{EX\#} \hat{e}x_2 = \{ \hat{e}x : n \mapsto \hat{s}t \mid \hat{s}t = \begin{cases} \hat{e}x_1(n) & \text{if } n \in \text{dom}(\hat{e}x_1) \setminus \text{dom}(\hat{e}x_2) \\ \hat{e}x_2(n) & \text{if } n \in \text{dom}(\hat{e}x_2) \setminus \text{dom}(\hat{e}x_1) \\ \hat{e}x_1(n) \nabla_{ST\#} \hat{e}x_2(n) & \text{if } n \in \text{dom}(\hat{e}x_1) \cap \text{dom}(\hat{e}x_2) \end{cases} \}$$

- Stati:  $\hat{s}t_1 \nabla_{ST\#} \hat{s}t_2 = (\hat{e}_1 \nabla_{E\#} \hat{e}_2, \hat{s}_1 \nabla_{S\#} \hat{s}_2, \hat{e}x_1 \nabla_{EX\#} \hat{e}x_2)$

# Teoremi

E' stato provato che:

- $\gamma_{ST}$  è monotona:  
 $\forall \hat{st}_1, \hat{st}_2 \in ST^\#, \hat{st}_1 \leq_{ST} \hat{st}_2 : \gamma_{ST}(\hat{st}_1) \sqsubseteq_{ST} \gamma_{ST}(\hat{st}_2)$
- $\alpha_{ST}$  è monotona:  
 $\forall ST_1, ST_2 \subseteq ST, ST_1 \subseteq ST_2 : \alpha_{ST}(ST_1) \leq_{ST} \alpha_{ST}(ST_2)$
- $ST' \sqsubseteq_{ST} \gamma_{ST}(\alpha_{ST}(ST'))$
- $\hat{st} = \alpha_{ST}(\gamma_{ST}(\hat{st}))$
- $\sqcup_{ST}$  è un operatore di least upper bound
- $\sqcap_{ST}$  è un operatore di greatest lower bound
- $\nabla_{ST}$  è un operatore di widening

# Proprietà

Per l'analisi di una proprietà abbiamo inserito il concetto di firewall astratto. Tale firewall (su cui è parametrizzata la nostra analisi) interviene quando:

- si accede ad un campo di un oggetto
- si invoca un metodo di un oggetto

Tale firewall analizza l'oggetto e cosa si accede, e in base a ciò può intervenire lanciando (nell'analisi astratta) un'eventuale eccezione. Esempio: nel corso dell'analisi della proprietà di applet isolation, si invoca un metodo di un oggetto. Il firewall controlla se il package da cui si accede l'oggetto è lo stesso dell'oggetto acceduto. Se così non è, controlla se il metodo invocato è condivisibile. Se non è così, non invoca il metodo e inserisce il lancio dell'eccezione all'interno dell'analisi astratta.

# Analisi statica modulare di linguaggi ad oggetti

L'analizzatore è stato implementato a partire dai concetti proposti e sviluppati nella tesi di dottorato di Francesco Logozzo ("Modular static analysis of object-oriented languages", 15/06/2004, Ecole Polytechnique, Paris).

Semantica concreta di una classe: l'insieme di tutti i possibili stati degli oggetti che possono essere istanziati a partire da questa.

Un'approssimazione sound di tale semantica concreta è data dalla soluzione del seguente sistema di equazioni.

$$s_0 = \bigsqcup_{init \in Constructors} M[[init]](\emptyset, \top_{init})$$

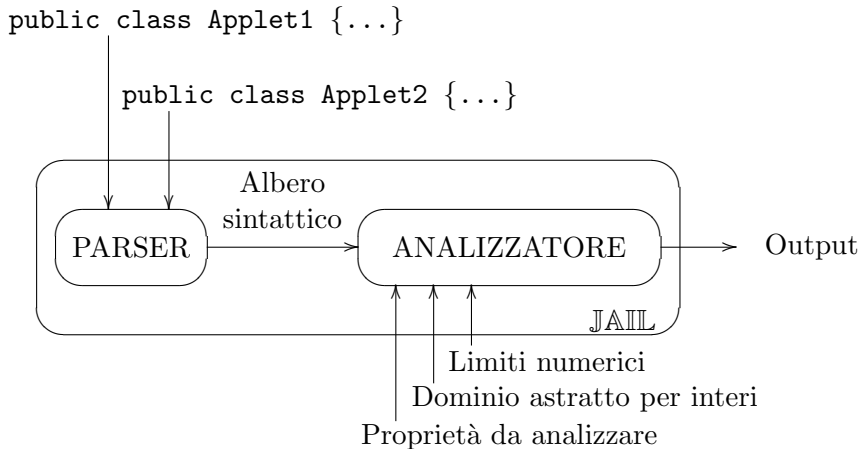
$$s_i = s_{i-1} \sqcup_{ST} \bigsqcup_{me \in Methods} M[[me]](s_{i-1}, \top_{me}) \quad \text{if } i \in [1..lci]$$

$$s_i = s_{i-1} \nabla_{ST} \bigsqcup_{me \in Methods} M[[me]](s_{i-1}, \top_{me}) \quad \text{if } i > [lci]$$

dove:

- $M[[me]]$  rappresenta la semantica del metodo  $me$ , a partire da uno stato iniziale e dal valore dei parametri del metodo
- $Constructors$  è l'insieme dei costruttori della classe analizzata
- $Methods$  è l'insieme dei metodi della classe analizzata
- $\top_{me}$  contiene la lista dei parametri del metodo  $me$  ciascuno con valore  $\top$

# Struttura del tool



## Screenshot

Abstract domain (int/double): Sign

Abstract domain for objects: NullPointerException

Loop limit: 20

Recursion limit: 10

Class invariant limit: 5

Select an example: Airline

c:\java Load

Row	Column
11	1

```

package smart_card;

public class Use extends Applet{
    private int user;
    private Service service;

    public Use(int i) {
        if(i>0)
            user=1;
        else user=0;
        service=JCSYSTEM.getAppletSherableInterfaceObject(Servi
        service.init(user);
    }

    public int use() {
        return service.read();
    }

    public int use(int i) {
        service.write(i);
        return 1;
    }
}

```

Abstract result

- service
- Value returned by method use(int i)
- Value returned by method use()
- user

Analyse the class

# Risultati

- L'analizzatore è stato implementato e può essere provato all'url <http://www.dsi.unive.it/ferrara/jail>
- Sono stati implementati vari domini astratti numerici (intervalli, segni, congruenze, ...) e alcune proprietà (assenza di NullPointerException, rispetto della proprietà di applet isolation di JavaCard, ...)
- In particolare è stata sviluppata nel dettaglio l'analisi della proprietà di applet isolation

# Future works

- Estensione dell'analizzatore ad altre proprietà, tra cui l'assenza di nested transactions, proprietà definita dal JavaCard Runtime Environment
- Progettazione e implementazione di un dominio astratto per valori numerici relazionale (dominio ottagonale)
- Implementazione della semantica completa delle librerie Java Card