History based flow analysis in the lambda calculus

Tomasz Blanc  
Jean-Jacques Lévy  

INRIA Rocquencourt and MSR-INRIA Joint Centre

November 13, 2007
Plan

1. Motivations
2. $\lambda$-calculus, principals and independence
3. $\lambda$-calculus and the Chinese Wall
4. Future works
Motivations
Restricting rights of downloaded programs is not sufficient...
Restricting rights of downloaded programs is not sufficient...
Restricting rights of downloaded programs is not sufficient...
Restricting rights of downloaded programs is not sufficient...
Restricting rights of downloaded programs is not sufficient...
... since attackers can borrow privileges from local programs [Hardy].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.

```
Effacer_fichier
```
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].

```
Navigateur
Efface_fichier_TMP
"PlugIn.tmp"
```

```
Efface_fichier_TMP
Navigateur
```

"PlugIn.tmp"
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].

```
Navigateur
Efface_fichier_TMP
Efface_fichier
"PlugIn.tmp"
```
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].

```
Efface_fichier
Navigateur
Efface_fichier_TMP
"PlugIn.tmp"
```
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet–Gordon].
First approach: stack inspection

- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem: there remains (indirect) ways of acting outside function calls [Fournet-Gordon].
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.

Non-interference: public output does not rely on secret inputs. Static analysis is doable even on complete languages (FlowCaml, JIF).
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.

- Non-interference: public output does not rely on secret inputs.

- Static analysis is doable even on complete languages (FlowCaml, JIF).
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
Second approach: Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.

Efface_fichier
"mon_fichier"
Efface_fichier_TMP
Navigateur
Second approach : Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
- **Non-interference** : public output does not rely on secret inputs.
Second approach : Information Flow

- Data are classified in several categories and their propagation is tracked during program execution.
- **Non-interference** : public output does not rely on secret inputs.
- Static analysis is do-able even on complete languages (FlowCaml, JIF).
Third approach : the Chinese Wall

- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.

- Charlie may interact with Alice and Bob.
Third approach: the Chinese Wall

- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.

Charlie may interact with Alice and Bob.
But as soon as Charlie interacts with Alice...
Third approach: the Chinese Wall

- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.

Charlie may interact with Alice and Bob.
But as soon as Charlie interacts with Alice, Charlie may no longer interact with Bob.
Third approach: the Chinese Wall

- Conflicts of interest in "economy" [Brewer-Nash].
- Alice and Bob compete for a contract; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.

Charlie may interact with Alice and Bob.
- But as soon as Charlie interacts with Alice, Charlie may no longer interact with Bob.
### Summary

<table>
<thead>
<tr>
<th>Safety policy</th>
<th>Safety property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Inspection</td>
<td>-</td>
</tr>
<tr>
<td>Flow Information</td>
<td>Non interference</td>
</tr>
<tr>
<td>Chinese Wall</td>
<td>?</td>
</tr>
</tbody>
</table>

**Objectives:**
- define the Chinese Wall in the $\lambda$-calculus.
- examine the safety property of the Chinese Wall policy.
Summary

<table>
<thead>
<tr>
<th>Safety policy</th>
<th>Safety property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Inspection</td>
<td>-</td>
</tr>
<tr>
<td>Flow Information</td>
<td>Non interference</td>
</tr>
<tr>
<td>Chinese Wall</td>
<td>?</td>
</tr>
</tbody>
</table>

Objectives:

- define the Chinese Wall in the \( \lambda \)-calculus.
- examine the safety property of the Chinese Wall policy.
\[\lambda\]-calculus, principals and independence
Alice, Bob, Charlie are principals.

\[ A, B, \ldots \]

Terms of \( \lambda_n \)-calculus:

\[ M, N ::= x \text{ Variable} \]
\[ \ | \ (\lambda x. M)^A \text{ Abstraction} \]
\[ \ | \ (MN)^A \text{ Application} \]

Values:

\[ V ::= (\lambda x. M)^A \]

Remark: principals differ from labels in the labelled \( \lambda \)-calculus.
\( \lambda_n \)-calculus: a \( \lambda \)-calculus with principals

- Alice, Bob, Charlie are principals.
  
  \( A, B, \ldots \)

- Terms of \( \lambda_n \)-calculus:
  
  \[
  M, N ::= \ x \quad \text{Variable} \\
  \quad \mid (\lambda x. M)^A \quad \text{Abstraction} \\
  \quad \mid (MN)^A \quad \text{Application}
  \]

- Values:
  
  \[
  V ::= (\lambda x. M)^A
  \]

- **Remark**: principals differ from labels in the labelled \( \lambda \)-calculus.
Reduction in $\lambda_n$-calculus

$$(\beta) \quad ((\lambda x.M)^A N)^B \rightarrow M\{x\backslash N\}$$
An example of reduction in the $\lambda_n$-calculus

\[
(((\lambda x. (\lambda y. y)^C)^C z)^A z)^B
\]
An example of reduction in the $\lambda^n$-calculus

\[
(((\lambda x.(\lambda y.y)^C)^C z)^A z)^B \quad \rightarrow \quad ((\lambda y.y)^C z)^B
\]
An example of reduction in the $\lambda_n$-calculus

$((\lambda x. (\lambda y. y)^C)^C z)^A z)^B \rightarrow ((\lambda y. y)^C z)^B$
An example of reduction in the $\lambda_n$-calculus

$$(((\lambda x. (\lambda y.y)^C)C z)^A z)^B \rightarrow (\lambda y.y)^C z)^B \rightarrow z$$
Basic properties of the $\lambda_n$-calculus

- Confluence
- Finite Developments
- Standardisation
Definition

The reduction \( M \xrightarrow{((\lambda x. N)^B P)^C} M' \) ignores A iff \( A \notin \{B, C\} \).

- Also written \( M \xrightarrow{\neg A} M' \).
- We write \( M \xrightarrow{\neg A} M' \) if every reduction step ignores A.
Reduction ignoring a principal

**Definition**

The reduction $M \xrightarrow{((\lambda x. N)B P)C} M'$ ignores $A$ iff $A \not\in \{B, C\}$.

- Also written $M \xrightarrow{\neg A} M'$.
- We write $M \xrightarrow{\neg A} \neg A M'$ if every reduction step ignores $A$.

**Example:**

```
@B
@A
@A
@C
@C
@C
@C
@C
@C
@C
```
Reduction ignoring a principal

Definition

The reduction $M \xrightarrow{((\lambda x. N)^B P)^C} M'$ ignores $A$ iff $A \notin \{B, C\}$.

- Also written $M \xrightarrow{\neg A} M'$.
- We write $M \xrightarrow{\neg A} M'$ if every reduction step ignores $A$.

Example:

```
\[
\lambda x^C z \lambda y^C y \, @A \, @B \xrightarrow{\neg B} \lambda y^C y \, @B \]
```

```
\[
\lambda x^C z \lambda y^C y \, @A \, @B \xrightarrow{\neg B} \lambda y^C y \, @B \]
```

```
\[
\lambda x^C z \lambda y^C y \, @A \, @B \xrightarrow{\neg B} \lambda y^C y \, @B \]
```

```
\[
\lambda x^C z \lambda y^C y \, @A \, @B \xrightarrow{\neg B} \lambda y^C y \, @B \]
```
**Reduction ignoring a principal**

**Definition**

The reduction \( M \xrightarrow{((\lambda x. N)^B P)^C} M' \) ignores \( A \) iff \( A \not\in \{B, C\} \).

- Also written \( M \xrightarrow{\neg A} M' \).
- We write \( M \xrightarrow{\neg A} M' \) if every reduction step ignores \( A \).

**Example:**

\[
\begin{array}{c}
\lambda x^C z \\downarrow \\
\lambda y^C y \\downarrow \\
y \\
\end{array}
\xrightarrow{\neg B}
\begin{array}{c}
\lambda y^C z \\downarrow \\
y \\downarrow \\
\end{array}
\xrightarrow{\neg A}
\begin{array}{c}
\lambda y^C z \\downarrow \\
\end{array}
\]

\[
\begin{array}{c}
\lambda x^C z \\downarrow \\
\lambda y^C y \\downarrow \\
y \\
\end{array}
\xrightarrow{\neg B}
\begin{array}{c}
\lambda y^C z \\downarrow \\
y \\downarrow \\
\end{array}
\xrightarrow{\neg A}
\begin{array}{c}
\lambda y^C z \\downarrow \\
\end{array}
\]

\[
\begin{array}{c}
\lambda x^C z \\downarrow \\
\lambda y^C y \\downarrow \\
y \\
\end{array}
\xrightarrow{\neg B}
\begin{array}{c}
\lambda y^C z \\downarrow \\
y \\downarrow \\
\end{array}
\xrightarrow{\neg A}
\begin{array}{c}
\lambda y^C z \\downarrow \\
\end{array}
\]

\[
\begin{array}{c}
\lambda x^C z \\downarrow \\
\lambda y^C y \\downarrow \\
y \\
\end{array}
\xrightarrow{\neg B}
\begin{array}{c}
\lambda y^C z \\downarrow \\
y \\downarrow \\
\end{array}
\xrightarrow{\neg A}
\begin{array}{c}
\lambda y^C z \\downarrow \\
\end{array}
\]

\[
\begin{array}{c}
\lambda x^C z \\downarrow \\
\lambda y^C y \\downarrow \\
y \\
\end{array}
\xrightarrow{\neg B}
\begin{array}{c}
\lambda y^C z \\downarrow \\
y \\downarrow \\
\end{array}
\xrightarrow{\neg A}
\begin{array}{c}
\lambda y^C z \\downarrow \\
\end{array}
\]
Actions of $A$ and $B$ are \textit{independent} if they commute.
• Actions of $A$ and $B$ are independent if they commute.
Independence

- Actions of $A$ and $B$ are independent if they commute.
Independence

- Actions of $A$ and $B$ are independent if they commute.
Independence

- Actions of $A$ and $B$ are independent if they commute.
Actions of $A$ and $B$ are independent if they commute.
Acts of $A$ and $B$ are independent if they commute.
Independence

Actions of $A$ and $B$ are **independent** if they commute.
Independence

- Actions of $A$ and $B$ are **independent** if they commute.
Actions of $A$ and $B$ are independent if they commute.
Independence

Definition (Independence)

The reduction $R : M \rightarrow N$ is **independent** of the interaction between $A$ and $B$ iff there exists $R_A : M \xrightarrow{\gamma A} M_A$ and $R_B : M \xrightarrow{\gamma B} M_B$ such that $R \leq R'$ (i.e. $R/R'$ is empty) with $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$.
Definition (Independence)

The reduction $R : M \rightarrow N$ is **independent** of the interaction between $A$ and $B$ iff there exists $R_A : M \xrightarrow{\neg A} M_A$ and $R_B : M \xrightarrow{\neg B} M_B$ such that $R \leq R'$ (i.e. $R/R'$ is empty) with $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$.
Definition (Independence)

The reduction $R : M \rightarrow N$ is independent of the interaction between $A$ and $B$ iff there exists $R_A : M \xrightarrow{\neg A} M_A$ and $R_B : M \xrightarrow{\neg B} M_B$ such that $R \leq R'$ (i.e. $R/R'$ is empty) with $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$.
Definition (Independence)

The reduction $R : M \rightarrow N$ is independent of the interaction between $A$ and $B$ iff there exists $R_A : M \xrightarrow{\neg A} M_A$ and $R_B : M \xrightarrow{\neg B} M_B$ such that $R \leq R'$ (i.e. $R/R'$ is empty) with $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$. 

![Independence Diagram]
**Independence**

**Definition (Independence)**

The reduction $R : M \rightarrow N$ is independent of the interaction between $A$ and $B$ iff there exists $R_A : M \xrightarrow{\neg A} M_A$ and $R_B : M \xrightarrow{\neg B} M_B$ such that $R \leq R'$ (i.e. $R/R'$ is empty) with $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$.
This reduction is not independent of the interaction between $A$ and $B$. 
This reduction is independent of the interaction between $A$ and $B$. 
This reduction is independent of the interaction between $A$ and $B$. 
The $\lambda_n$-calculus : summary

- A $\lambda$-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the $\lambda_n$-calculus?
  - This policy relies on history.
  - We use the labelled $\lambda$-calculus to track history of interactions.
- Which safety property is guaranteed by the Chinese Wall policy?
  - We show that a reduction following the Chinese Wall policy between $A$ and $B$ is independent of the interaction between $A$ and $B$. 
The $\lambda_n$-calculus: summary

- A $\lambda$-calculus with **principals**.
- A safety property: **independence**.
- How to express the Chinese Wall policy in the $\lambda_n$-calculus?
  - This policy relies on history.
    - We use the labelled $\lambda$-calculus to track history of interactions.
- Which safety property is guaranteed by the Chinese Wall policy?
  - We show that a reduction following the Chinese Wall policy between $A$ and $B$ is independent of the interaction between $A$ and $B$. 
The $\lambda_n$-calculus: summary

- A $\lambda$-calculus with principals.
- A safety property: independence.
- How to express the Chinese Wall policy in the $\lambda_n$-calculus?
  - This policy relies on history.
  - We use the labelled $\lambda$-calculus to track history of interactions.
- Which safety property is guaranteed by the Chinese Wall policy?
  - We show that a reduction following the Chinese Wall policy between $A$ and $B$ is independent of the interaction between $A$ and $B$. 
The $\lambda_n$-calculus: summary

- A $\lambda$-calculus with principals.
- A safety property: independence.
- How to express the Chinese Wall policy in the $\lambda_n$-calculus?
  - This policy relies on history.
  - We use the labelled $\lambda$-calculus to track history of interactions.
- Which safety property is guaranteed by the Chinese Wall policy?
  - We show that a reduction following the Chinese Wall policy between $A$ and $B$ is independent of the interaction between $A$ and $B$. 
The $\lambda_n$-calculus: summary

- A $\lambda$-calculus with principals.
- A safety property: independence.
- How to express the Chinese Wall policy in the $\lambda_n$-calculus?
  - This policy relies on history.
  - We use the labelled $\lambda$-calculus to track history of interactions.
- Which safety property is guaranteed by the Chinese Wall policy?
  - We show that a reduction following the Chinese Wall policy between $A$ and $B$ is independent of the interaction between $A$ and $B$. 
\( \lambda \)-calculus

and

the Chinese Wall
The labelled $\lambda_n$-calculus

Terms

\[ M, N ::= x \]
\[ \quad | \quad (\lambda x. N)^A \]
\[ \quad | \quad (MN)^A \]
\[ \quad | \quad a : M \]

Atomic labels

\[ a, b ::= [\alpha] \mid [\alpha] \]

Compound labels

\[ \alpha, \beta ::= A a_1 a_2 \cdots a_n B \quad n \geq 0 \]

Values

\[ V, W ::= (\lambda x. N)^A \mid a : V \]
Labelled reduction

\[
\lambda x^B M \xrightarrow{\alpha} M \{x \setminus [\alpha] : N\}
\]

\[
R = (a_1 : \ldots : a_n : (\lambda x. M)^B N)^A \rightarrow [\alpha] : M\{x \setminus [\alpha] : N\}
\]

\[
\alpha = Aa_1 \ldots a_n B
\]

The redex name is \(\text{name}(R) = \alpha\).
Labelled reduction : an example

$$(((\lambda x.(\lambda y.y)^C)^Az)^Cz)^B$$
Labelled reduction: an example

\[
(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B \rightarrow ([CA] : (\lambda y.y)^C z)^B
\]
Labelled reduction: an example

\[
(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B \rightarrow ([CA]:(\lambda y.y)^C z)^B
\]
Labelled reduction: an example

\[
(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B \rightarrow ([CA] : (\lambda y.y)^C z)^B \\
\]
Independence and labels

- **Head sequence**: \( \tau(x) = \tau((\lambda x. M)^A) = \tau((MN)^A) = 0 \)
  \[ \tau(a : M) = a\tau(M) \]
Independence and labels

- Head sequence: \( \tau(x) = \tau((\lambda x. M)^A) = \tau((MN)^A) = 0 \)
  \( \tau(a : M) = a \tau(M) \)

  - Example: \( \tau(a : b : c : (\lambda x. x)^A) = abc \)
Independence and labels

- Head sequence: \( \tau(x) = \tau((\lambda x. M)^A) = \tau((MN)^A) = 0 \)
  \( \tau(a : M) = a\tau(M) \)

- Principals contained in atomic or compound labels:
  \[
  \Princ(Aa_1\ldots a_nB) = \{A, B\} \cup_{1 \leq i \leq n} \Princ(a_i) \\
  \Princ([\alpha]) = \Princ([\alpha]) = \Princ(\alpha)
  \]
Independence and labels

- **Head sequence**: \( \tau(x) = \tau((\lambda x. M)^A) = \tau((MN)^A) = 0 \)
  \( \tau(a : M) = a\tau(M) \)

- **Principals contained in atomic or compound labels**: 
  \[ \text{Princ}(Aa_1 \ldots a_nB) = \{A, B\} \cup _{1\leq i \leq n} \text{Princ}(a_i) \]
  \[ \text{Princ}(\lceil \alpha \rceil) = \text{Princ}(\lfloor \alpha \rfloor) = \text{Princ}(\alpha) \]
  
Independence and labels

- **Head sequence**: \( \tau(x) = \tau((\lambda x. M)^A) = \tau((MN)^A) = 0 \)
  \[ \tau(a : M) = a\tau(M) \]

- **Principals contained in atomic or compound labels**:
  \[
  \text{Princ}(Aa_1 \ldots a_nB) = \{A, B\} \cup_{1 \leq i \leq n} \text{Princ}(a_i)
  \]
  \[
  \text{Princ}([\alpha]) = \text{Princ}(\lfloor \alpha \rfloor) = \text{Princ}(\alpha)
  \]

**Definition (Separation)**

A sequence of atomic labels \( a_1 \ldots a_n \) separates the principals \( A \) and \( B \) iff, for every \( 1 \leq i \leq n \), we have \( \{A, B\} \not\subseteq \text{Princ}(a_i) \).
Independence and labels

- **Head sequence:** \( \tau(x) = \tau((\lambda x.M)^A) = \tau((MN)^A) = 0 \)
  \[ \tau(a : M) = a \tau(M) \]

- **Principals contained in atomic or compound labels:**
  \[ \text{Princ}(Aa_1...a_nB) = \{A, B\} \cup_{1 \leq i \leq n} \text{Princ}(a_i) \]
  \[ \text{Princ}(\lceil \alpha \rceil) = \text{Princ}([\alpha]) = \text{Princ}(\alpha) \]

**Definition (Separation)**

A sequence of atomic labels \( a_1 \ldots a_n \) separates the principals \( A \) and \( B \) iff, for every \( 1 \leq i \leq n \), we have \( \{A, B\} \not\subseteq \text{Princ}(a_i) \).

**Examples:**
- \[ [AC][C[DE]B] \] separates \( A \) et \( B \).
- \[ [DC][C[AE]B] \] does not separate \( A \) et \( B \).
Independence and labels : separation

Theorem (Separation)

*If M is an unlabelled term and if the reduction $M \rightarrow V$ is independent of the interaction between A and B, then $\tau(V)$ separates A and B.*
Theorem (Separation)

If $M$ is an unlabelled term and if the reduction $M \rightarrow V$ is independent of the interaction between $A$ and $B$, then $\tau(V)$ separates $A$ and $B$. 

\[
\begin{array}{c}
\lambda x^C \quad \lambda y^C \\
A \\
@ \\
y \quad u
\end{array}
\rightarrow
\begin{array}{c}
\lambda x^C \quad [AC]^u \\
\quad \lambda u^C \\
B \\
@ \\
z
\end{array}
\rightarrow
\begin{array}{c}
\quad [BC] \\
\quad [AC] \\
\quad [AC] \\
\lambda u^C \\
\quad u
\end{array}
\]
Theorem (Separation)

If \( M \) is an unlabelled term and if the reduction \( M \rightarrow V \) is independent of the interaction between \( A \) and \( B \), then \( \tau(V) \) separates \( A \) and \( B \).
Independence and labels: separation

Theorem (Separation)

If $M$ is an unlabelled term and if the reduction $M \rightarrow V$ is independent of the interaction between $A$ and $B$, then $\tau(V)$ separates $A$ and $B$.

The head sequence $[BC][AC][AC]$ separates $A$ and $B$. 
Independence and separation

**Theorem**

If $M \rightarrow V$ and if $\tau(V)$ separates $A$ and $B$, then there is a reduction $\mathcal{R} : M \rightarrow W$ independent of the interaction between $A$ and $B$. 

⋆ The label $\lceil AA \rceil$ separates $A$ and $B$.

⋆ This reduction is not independent of the interaction between $A$ and $B$. 
Theorem

If $M \rightarrow V$ and if $\tau(V)$ separates $A$ and $B$, then there is a reduction $\mathcal{R} : M \rightarrow W$ independent of the interaction between $A$ and $B$. 
Independence and separation

Theorem

If $M \rightarrow V$ and if $\tau(V)$ separates $A$ and $B$, then there is a reduction $R : M \rightarrow W$ independent of the interaction between $A$ and $B$.

★ The label $[AA]$ separates $A$ et $B$. 
Independence and separation

**Theorem**

*If* $M \rightarrow V$ *and if* $\tau(V)$ *separates* $A$ *and* $B$, *then there is a reduction* $\mathcal{R} : M \rightarrow W$ *independent of the interaction between* $A$ *and* $B$.

- The label $\langle AA \rangle$ separates $A$ and $B$.
- This reduction *is not* independent of the interaction between $A$ and $B.
Independence and separation

Theorem

If $M \rightarrow V$ and if $\tau(V)$ separates $A$ and $B$, then there is a reduction $R : M \rightarrow W$ independent of the interaction between $A$ and $B$.

★ The label $[AA]$ separates $A$ et $B$.

★ This reduction is independent of the interaction between $A$ and $B$. 
Expressing Chinese Wall in the $\lambda_n$-calculus

- The Chinese Wall between $A$ and $B$ is written $CW(A, B)$.

**Definition (Chinese Wall)**

A reduction follows $CW(A, B)$ iff every redex $R$ contracted by this reduction is such that:

$$\{A, B\} \not\subseteq \text{Princ}(\text{name}(R))$$
Chinese Wall in the $\lambda_n$-calculus: example 1/2

Princ(name($R_1$)) = \{A, C\}
Princ(name($R_2$)) = \{A, B, C\}

This reduction does not follow $\mathcal{CW}(A, B)$. 
Chinese Wall in the $\lambda_n$-calculus: example 2/2

\[ \Princ(name(R_1)) = \{ A, C \} \]
\[ \Princ(name(R_2)) = \{ B, C \} \]

This reduction follows $\mathcal{CW}(A, B)$. 
Correction of $\mathcal{CW}(A, B)$

Theorem (Correction)

If $R : M \rightarrow N$ follows $\mathcal{CW}(A, B)$, then $R$ is independent of the interaction between $A$ and $B$.

The Chinese Wall guarantees the independence.
Correction of $CW(A, B)$: example

The reduction follows $CW(A, B)$...
Correction of $\mathcal{CW}(A, B)$: example

...hence it is independent of the interaction between $A$ and $B$
Correction of $\mathcal{CW}(A, B)$ : proof

- Sublabel of a compound label :

  \[
  \alpha \leq \alpha \\
  \alpha \leq Aa_1 \ldots a_n B \text{ si } \exists i . \ a_i = [\beta] \text{ and } \alpha \leq \beta \\
  \alpha \leq Aa_1 \ldots a_n B \text{ si } \exists i . \ a_i = [\beta] \text{ and } \alpha \leq \beta
  \]

- Example : $\alpha \leq A[\alpha][\gamma]B$
Correction of $\mathcal{CW}(A, B)$: proof

- Sublabel of a compound label:
  \[
  \alpha \preceq \alpha \\
  \alpha \preceq Aa_1 \ldots a_nB \text{ si } \exists i \ . \ a_i = [\beta] \text{ and } \alpha \preceq \beta \\
  \alpha \preceq Aa_1 \ldots a_nB \text{ si } \exists i \ . \ a_i = [\beta] \text{ and } \alpha \preceq \beta
  \]

- Example: $\alpha \preceq A[\alpha][\gamma]B$
Correction of $\mathcal{CW}(A, B)$: proof

$M \xrightarrow{S_1} \xrightarrow{S_2} \xrightarrow{S_3} \ldots \xrightarrow{S_n} N$

$\quad R$
Correction of $\mathcal{CW}(A, B)$ : proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.
We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.
Correction of $\mathcal{CW}(A, B)$ : proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.
We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$. 
Correction of $CW(A, B)$ : proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.
We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$. 
Correction of $\mathcal{CW}(A, B)$ : proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$. We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

**Lemma (Completion)**

If $R : M \xrightarrow{S_1} \ldots \xrightarrow{S_n} N$ and if for every $i$, we have $\text{name}(S_i) = \alpha_i$, then $R_1 : M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N_1$ and $R \leq R_1$. 
Correction of $CW(A, B)$: proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$. We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

**Lemma (Completion)**

If $R : M \xrightarrow{S_1} \ldots \xrightarrow{S_n} N$ and if for every $i$, we have $\text{name}(S_i) = \alpha_i$, then $R_1 : M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N_1$ and $R \leq R_1$. 

\[ M \xrightarrow{S_1} S_2 \xrightarrow{S_3} \ldots \xrightarrow{S_n} N \]
\[ R \]
\[ R_1 \]
\[ R/R_1 = \emptyset^n \]
\[ R_1/R \]
\[ M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N_1 \]
Correction of $\mathcal{CW}(A, B)$: proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.
We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

**Lemma (Completion)**

If $R : M \xrightarrow{S_1} \ldots \xrightarrow{S_n} N$ and if for every $i$, we have $\text{name}(S_i) = \alpha_i$, then $R_1 : M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N_1$ and $R \leq R_1$. 
Correction of $\mathcal{CW}(A, B)$: proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$. We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Reordering)

If $R : M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N$, there is a reduction $R' : M \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_m} N'$ such that

1. $\{\beta_i\}_{1 \leq i \leq m} \subseteq \{\alpha_i\}_{1 \leq i \leq n}$
2. if $i < j$, then $\beta_j \nleq \beta_i$
3. $R \leq R'$
Correction of $C\mathcal{W}(A, B)$: proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.
We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Reordering)

*If $R : M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} N$, there is a reduction $R' : M \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_m} N'$ such that*

1. $\{\beta_i\}_{1 \leq i \leq m} \subseteq \{\alpha_i\}_{1 \leq i \leq n}$
2. if $i < j$, then $\beta_j \not\preceq \beta_i$
3. $R \leq R'$
Correction of $\mathcal{CN}(A, B)$: proof

If $i < j$, we have $\beta_j \not≺ \beta_i$.

- $\{\gamma_i\}_{1 \leq i \leq k}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) = \emptyset$.
- $\{\delta_i\}_{1 \leq i \leq k'}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) \neq \emptyset$.
- If $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$, if $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not≺ \beta_j$. 

```latex
def M \xrightarrow{S_1} S_2 \xrightarrow{S_3} \cdots \xrightarrow{S_n} N \\
\beta_1 \downarrow \\
\beta_2 \downarrow \\
\vdots \\
\beta_m \downarrow \\
N' \\

M \\
\beta_1 \downarrow \\
\beta_2 \downarrow \\
\vdots \\
\beta_m \downarrow \\
R'/R \\

R' \beta_3 \\
\vdots \\
R' \beta_3 \\
N'
```
Correction of $\mathcal{CW}(A, B)$: proof

If $i < j$, we have $\beta_j \not≺ \beta_i$.

• $\{\gamma_i\}_{1 \leq i \leq k}$ : elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) = \emptyset$.

• $\{\delta_i\}_{1 \leq i \leq k'}$ : elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) \neq \emptyset$.
Correction of $CW(A, B)$: proof

If $i < j$, we have $\beta_j \not\prec \beta_i$.

$\{\gamma_i\}_{1 \leq i \leq k}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) = \emptyset$.

$\{\delta_i\}_{1 \leq i \leq k'}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) \neq \emptyset$.

If $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$, if $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not\prec \beta_j$. 
Correction of $CW(A, B)$ : proof

If $i < j$, we have $\beta_j \not\preceq \beta_i$.

$\{\gamma_i\}_{1 \leq i \leq k}$ : elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) = \emptyset$.

$\{\delta_i\}_{1 \leq i \leq k'}$ : elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \text{Princ}(\beta_i) \neq \emptyset$.

If $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$, if $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not\preceq \beta_j$. 
**Correction of \( CW(A, B) \): proof**

\[
\begin{array}{cccccccc}
M & S_1 & S_2 & S_3 & \cdots & S_n & N \\
\beta_1 & \downarrow & \beta_2 & \downarrow & \beta_3 & \downarrow & \cdots & \beta_m & \downarrow & \beta_1 \\
R & \downarrow & R & \downarrow & R & \downarrow & \cdots & R & \downarrow & R \\
R'/R & \downarrow & R'/R & \downarrow & R'/R & \downarrow & \cdots & R'/R & \downarrow & R'/R \\
N' & \downarrow & N' & \downarrow & N' & \downarrow & \cdots & N' & \downarrow & N' \\
\end{array}
\]

- If \( i < j \) and \( \beta_i \in \{\delta_i\}_{1 \leq i \leq k'} \) and \( \beta_j \in \{\gamma_i\}_{1 \leq i \leq k} \), we have \( \beta_i \not\prec \beta_j \) and \( \beta_j \not\prec \beta_i \).

**Lemma (Permutation)**

If \( \alpha \not\prec \beta \) and \( \beta \not\prec \alpha \) and if \( R_1 : M \Rightarrow \Rightarrow N \), then we have \( R_2 : M \Rightarrow \Rightarrow N \) and \( R_1 \sim R_2 \).
Correction of $C\mathcal{W}(A, B)$ : proof

If $i < j$ and $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$ and $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not\prec \beta_j$ and $\beta_j \not\prec \beta_i$.

Lemma (Permutation)

If $\alpha \not\prec \beta$ and $\beta \not\prec \alpha$ and if $R_1 : M \Rightarrow \Rightarrow N$, then we have $R_2 : M \Rightarrow \Rightarrow N$ and $R_1 \sim R_2$. 
Correction of $\mathcal{CW}(A, B)$: proof

If $i < j$ et $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$ and $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not\prec \beta_j$ et $\beta_j \not\prec \beta_i$.

Lemma (Permutation)

If $\alpha \not\prec \beta$ and $\beta \not\prec \alpha$ and if $R_1 : M \xRightarrow{\alpha} \xRightarrow{\beta} N$, then we have $R_2 : M \xRightarrow{\beta} \xRightarrow{\alpha} N$ and $R_1 \sim R_2$. 
Correction of $\mathcal{C W}(A, B)$: proof

\begin{itemize}
  \item $\{\eta_i\}_{1 \leq i \leq p}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{A\}$.
  \item $\{\theta_i\}_{1 \leq i \leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{B\}$.
  \item For every $i, j$, we have $\eta_i \not\prec \theta_j$ and $\theta_j \not\prec \eta_i$.
\end{itemize}
Correction of $\mathcal{C}\mathcal{W}(A, B)$: proof

- $\{\eta_i\}_{1 \leq i \leq p}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{A\}$.
- $\{\theta_i\}_{1 \leq i \leq p'}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{B\}$.
- For every $i, j$, we have $\eta_i \not\prec \theta_j$ and $\theta_j \not\prec \eta_i$. 
Correction of $\mathcal{CW}(A, B)$ : proof

\[ M \xrightarrow{S_1} S_2 \xrightarrow{S_3} \ldots \xrightarrow{S_n} N \]

\[ \beta_1 \quad \beta_2 \quad \beta_3 \quad \ldots \quad \beta_m \]

\[ R' \]

\[ R'/R \]

\[ M_B \]

\[ \gamma_1 \quad \gamma_k \quad \eta_1 \quad \eta_p \quad \theta_1 \quad \theta_{p'} \]

\[ \{ \eta_i \}_{1 \leq i \leq p} : \text{elements of } \{ \delta_i \}_i \]
\[ \text{such that } \{ A, B \} \cap \text{Princ}(\delta_i) = \{ A \} \]

\[ \{ \theta_i \}_{1 \leq i \leq p'} : \text{elements of } \{ \delta_i \}_i \]
\[ \text{such that } \{ A, B \} \cap \text{Princ}(\delta_i) = \{ B \} \]

\[ \text{For every } i, j, \text{ we have } \eta_i \not\preceq \theta_j \text{ and } \theta_j \not\preceq \eta_i \]
Correction of $C\mathcal{W}(A, B)$ : proof

$M \xrightarrow{S_1} S_2 \xrightarrow{S_3} \cdots \xrightarrow{S_n} N$

$M \xrightarrow{\gamma_1} R \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_k} R' \xrightarrow{\theta_1} R'/R \xrightarrow{\eta_1} \cdots \xrightarrow{\eta_p} N' \xrightarrow{\theta_{p'}} M_B$

- $\{\eta_i\}_{1 \leq i \leq p}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{A\}$.
- $\{\theta_i\}_{1 \leq i \leq p'}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{B\}$.
- For every $i, j$, we have $\eta_i \not\prec \theta_j$ and $\theta_j \not\prec \eta_i$. 
Correction of $\mathcal{C}(A, B)$: proof

$\{\eta_i\}_{1 \leq i \leq p}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{A\}$.

$\{\theta_i\}_{1 \leq i \leq p'}$ : elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{B\}$.

For every $i, j$, we have $\eta_i \not\prec \theta_j$ and $\theta_j \not\prec \eta_i$. 

$\gamma_1 \beta_1 \gamma_k \eta_1 \eta_p \theta_1 \beta_1 \beta_2 \eta_1 \beta_m \theta_p \eta_p$
Correction of $CW(A, B)$: proof

\[ \begin{array}{c}
M \xrightarrow{S_1} S_2 \xrightarrow{S_3} \cdots \xrightarrow{S_n} N \\
M_A \xleftarrow{M_B} R_A \xleftarrow{R_B} N' \xleftarrow{R_A/R_B} R_B/R_A \xleftarrow{M_A} R_A \xleftarrow{R_B} N' \xleftarrow{M_B} R_B
\end{array} \]
Correction of $\mathcal{CW}(A, B)$: proof

\[ \begin{array}{cccccc}
M & \xrightarrow{S_1} & S_2 & \xrightarrow{S_3} & \cdots & \xrightarrow{S_n} N \\
\downarrow \beta_1 & & & & & \downarrow R \\
R' & \xrightarrow{\beta_2} & R' \xrightarrow{\beta_3} & \cdots & \xrightarrow{\beta_m} N' \\
\downarrow \beta_m & & & & & \downarrow R'/R \\
N' & & & & & \\
\end{array} \]
### \(\lambda\)-calculus and Chinese Wall: summary

1. Safety property: independence
2. Correspondence between labelled lambda calculus and independence

<table>
<thead>
<tr>
<th>Safety policy</th>
<th>Safety property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack inspection</td>
<td>-</td>
</tr>
<tr>
<td>Information flow</td>
<td>Non interference</td>
</tr>
<tr>
<td>Chinese Wall</td>
<td>Independence</td>
</tr>
</tbody>
</table>
Future works
Objectives

1. Static information flow in the $\lambda$-calculus
   - labelled $\lambda$-calculus and DCC [Riecke], FlowCaml as [Simonet, Pottier], DCC+ [Abadi], etc

2. Reduction strategies
   - call-by-value $\lambda$-calculus
   - weak $\lambda$-calculus

3. Adding delta rules
   - Imperative features and exceptions
   - Safety rules (safety operators: uses or binds)

4. Concurrent features
   - Permutation equivalence and Event structures
   - Reversible processes (backtracking) [Jean Krivine]
Conclusion: non interference

- Non interference: the labels of the \( \lambda \)-calculus express functional interference.
- In the \( \lambda \)-calculus with references, labels have to also capture interference with memory.
  - A memory cell interferes within some time interval.
  - We can use irreversibility of contexts in the labelled \( \lambda \)-calculus [Blanc].
Non interference: the labels of the $\lambda$-calculus express functional interference.

In the $\lambda$-calculus with references, labels have to also capture interference with memory.

- A memory cell interferes within some time interval.

- We can use irreversibility of contexts in the labelled $\lambda$-calculus [Blanc].
Conclusion: non interference

- Non interference: the labels of the $\lambda$-calculus express **functional interference**.
- In the $\lambda$-calculus with references, labels have to also capture interference with memory.
  - A memory cell interferes within some time **interval**.
  - We can use irreversibility of contexts in the labelled $\lambda$-calculus [Blanc].
Conclusion: independence

1. Created principals and extended independence.
2. Link between non-interference and independence: express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that $k$CFA is NP as soon as $k > 0$. [ICFP 07].
5. Simple proofs for safety properties.
Conclusion: independence

- Created principals and extended independence.
- Link between non-interference and independence: express these properties within a common framework.
- Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
- David Van Horn and Harry Mairson showed that $k$CFA is NP as soon as $k > 0$. [ICFP 07].
- Simple proofs for safety properties.
Conclusion: independence

1. Created principals and extended independence.
2. Link between non-interference and independence: express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that $k$CFA is NP as soon as $k > 0$. [ICFP 07].
5. Simple proofs for safety properties.
Conclusion: independence

1. Created principals and extended independence.
2. Link between non-interference and independence: express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that $k$CFA is NP as soon as $k > 0$. [ICFP 07].
5. Simple proofs for safety properties.
Conclusion: independence

1. Created principals and extended independence.
2. Link between non-interference and independence: express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that $k$CFA is NP as soon as $k > 0$. [ICFP 07].
5. Simple proofs for safety properties.