History based flow analysis in the lambda calculus

Tomasz Blanc
Jean-Jacques Lévy

INRIA Rocquencourt and MSR-INRIA Joint Centre

November 14, 2006
Plan

1. Motivations
2. $\lambda$-calculus, principals and independence
3. $\lambda$-calculus and the Chinese Wall
4. Future works
Motivations
Security and Programming languages

- 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.
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.

Efface_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.

Efface_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.
- 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].

\begin{itemize}
  \item \texttt{Efface\_fichier}\texttt{"PlugIn.tmp"}
  \item \texttt{Efface\_fichier\_TMP}
  \item \texttt{Navigateur}
\end{itemize}
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
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].

```
Efface_fichier
Navigateur
Efface_fichier_TMP
"mon_fichier"
```

[Diagram: Stack inspection diagram with arrows indicating function calls and a crossed-out symbol indicating an indirect way of acting outside function calls.]
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.

```javascript
Efface_fichier_TMP

Navigateur
```
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.

Navigateur
Efface_fichier_TMP
"mon_fichier"

Efface_fichier_TMP
Navigateur
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.
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.

![Diagram showing interactions between Alice, Bob, and Charlie.](image)
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.
Objectives:

- define the Chinese Wall in the $\lambda$-calculus.
- examine the safety property of the Chinese Wall policy.
\(\lambda\)-calculus, principals and independence
\( \lambda_n \)-calculus: a \( \lambda \)-calculus with principals

- Alice, Bob, Charlie are principals.
  
  \[ A, B, \ldots \]

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

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

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

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

$((\lambda x.(\lambda y.y)^C)^C_z)^A_z)^B \rightarrow ((\lambda y.y)^C_z)^B$
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
Reduction ignoring a principal

Definition

The reduction $M \xrightarrow{((\lambda x.N)^B)_C} M'$ 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{align*}
&@B \\
&@A \\
&@A \\
&\lambda x^C \quad Z \\
&\lambda y^C \quad Z \\
&y
\end{align*}
```
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 $M \xrightarrow{\not\lambda A} M'$.
- We write $M \xrightarrow{\not\lambda A} M'$ if every reduction step ignores $A$.

**Example:**

```
\[ \begin{array}{c}
\lambda x^C \quad \lambda y^C \\
\lambda y^C \\
y
\end{array} \xrightarrow{\not\lambda B} \begin{array}{c}
\lambda y^C \\
y
\end{array} \xrightarrow{\not\lambda B} \begin{array}{c}
\lambda y^C \\
y
\end{array} \]
```
Reduction ignoring a principal

**Definition**

The reduction $\phi$ \((\lambda x. N)^B_P)^C\) $\rightarrow M'$ 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 :**

```
λx^C z
  λy^C y
  y
λx^C z
  λy^C z
  y
```

```
λy^C z
  y
```

```
λy^C z
  y
  z
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```

```
λy^C z
  y
```

```
λx^C z
  λy^C z
  y
  z
```
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.
Actions of $A$ and $B$ are independent if they commute.
Actions of $A$ and $B$ are independent if they commute.
Independence

- Actions of $A$ and $B$ are \textit{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.
• Actions of $A$ and $B$ are independent if they commute.
Independence

- 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 \( R_A : M \xrightarrow{\neg A} M_A \) 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

Definition (Independence)

The reduction \( R : M \to N \) 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

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 $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$. 
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 $R' = R_A; (R_B/R_A) = R_B; (R_A/R_B)$. 

\[
\begin{array}{c}
M \\
\downarrow R \\
N \\
\downarrow \neg A \\
M_A \\
\downarrow R_B \\
M_B \\
\downarrow \neg B \\
N' \\
\downarrow R'/R \\
N' \\
\downarrow R_B/R_A \\
N' \\
\downarrow R_A/R_B \\
N'
\end{array}
\]
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 $B$. 
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
\(\lambda\)-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

\[ \alpha, \beta ::= Aa_1a_2 \cdots a_nB \quad n \geq 0 \]

Values

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

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

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

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

\[ (((\lambda x. (\lambda y. y)^C)^A z)^C z)^B \]

\[ (\lambda x. (\lambda y. y)^C)^A z)^C z \]

\[ B \]

\[ C \]

\[ z \]

\[ z \]
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

\[(\lambda x. (\lambda y. y)^C z)^A z)^C z^B \rightarrow ([CA] : (\lambda y. y)^C z)^B\]
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**:
  \[
  \text{Princ}(Aa_1\ldots a_nB) = \{A, B\} \cup_{1 \leq i \leq n} \text{Princ}(a_i) \\
  \text{Princ}([\alpha]) = \text{Princ}([\alpha]) = \text{Princ}(\alpha)
  \]
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**:
  \( \text{Princ}(Aa_1 \ldots a_nB) = \{A, B\} \cup \bigcup_{1 \leq i \leq n} \text{Princ}(a_i) \)
  \[ \text{Princ}(\lfloor \alpha \rfloor) = \text{Princ}([\alpha]) = \text{Princ}(\alpha) \]

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:
  \[ \text{Princ}(Aa_1...a_nB) = \{A, B\} \cup_{1 \leq i \leq n} \text{Princ}(a_i) \]
  \[ \text{Princ}([\alpha]) = \text{Princ}([\alpha]) = \text{Princ}(\alpha) \]

Definition (Separation)

A sequence of atomic labels \( a_1...a_n \) 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 \( B \).
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 $B$. 

The head sequence $\lambda^\uparrow BC \lambda^\uparrow AC \lambda^\downarrow AC$ 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 $B$. 
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 $R : M \rightarrow W$ independent of the interaction between $A$ and $B$. 

⋆ The label $\left\langle AA \right\rangle$ separates $A$ and $B$.

⋆ This reduction $R$ 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 $B$. 

\[ \lambda x^A \lambda y^B \lambda z^C @ \lambda z^C y^B u^{\langle AA \rangle} \rightarrow @ \lambda y^B \lambda z^C u^{\langle AA \rangle} \rightarrow @ \lambda y^B \lambda z^C u^{\langle B|AA|C \rangle} \]
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 \( R : M \rightarrow W \) independent of the interaction between \( A \) and \( B \).

- The label \([AA]\) separates \( A \) et \( B \).
- 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 $B$.

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

★ 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 $\text{CW}(A, B)$.
- If redex $R$ has name $Aa_1 \ldots a_nB$, then:
  - $A$ and $B$ interact (directly).
  - If $C \in \text{Princ}(a_i)$, then $C$ participated to the creation of this interaction.

**Definition (Chinese Wall)**

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

$$\{A, B\} \not\subseteq \text{Princ}(\text{name}(R))$$
Expressing Chinese Wall in the $\lambda_n$-calculus

- The Chinese Wall between $A$ and $B$ is written $\CW(A, B)$.
- If redex $R$ has name $A a_1 \ldots a_n B$, then:
  - $A$ and $B$ interact (directly).
  - If $C \in \text{Princ}(a_i)$, then $C$ participated to the creation of this interaction.

Definition (Chinese Wall)

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

$$\{A, B\} \not\subseteq \text{Princ(name}(R))$$
Expressing Chinese Wall in the $\lambda_n$-calculus

- The Chinese Wall between $A$ and $B$ is written $\text{CW}(A, B)$.
- If redex $R$ has name $Aa_1 \ldots a_nB$, then:
  - $A$ and $B$ interact (directly).
  - If $C \in \text{Princ}(a_i)$, then $C$ participated to the creation of this interaction.

**Definition (Chinese Wall)**

A reduction follows $\text{CW}(A, B)$ 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

$\lambda x^A y^C z \rightarrow \lambda y^C y^C [CA] z \rightarrow [B[CA]C] z$

\[
\begin{align*}
\text{Princ(name}(R_1)) &= \{A, C\} \\
\text{Princ(name}(R_2)) &= \{A, B, C\}
\end{align*}
\]

This reduction does not follow $C\lambda \forall (A, B)$. 
Chinese Wall in the $\lambda_n$-calculus: example 2/2

\[
\begin{align*}
\lambda x^C z \downarrow A \\
\lambda y^C z \\
\end{align*}
\]

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

This reduction follows $C\mathcal{W}(A, B)$. 
Theorem (Correction)

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

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

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

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

- **Sublabel of a compound label**:
  
  \[ \alpha \leq \alpha \]
  
  \[ \alpha \leq Aa_1 \ldots a_n B \text{ si } \exists i . \ a_i = \lceil \beta \rceil \text{ and } \alpha \leq \beta \]
  
  \[ \alpha \leq Aa_1 \ldots a_n B \text{ si } \exists i . \ a_i = \lfloor \beta \rfloor \text{ and } \alpha \leq \beta \]

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

- Sublabel of a compound label :

  \[ \alpha \leq \alpha \]
  \[ \alpha \leq Aa_1 \ldots a_nB \text{ si } \exists i . a_i = \lceil \beta \rceil \text{ and } \alpha \leq \beta \]
  \[ \alpha \leq Aa_1 \ldots a_nB \text{ si } \exists i . a_i = \lfloor \beta \rfloor \text{ and } \alpha \leq \beta \]

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

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

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

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

**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 $R \leq R_1$. 
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 $C\forall\forall(A, B)$: proof
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 $\mathcal{CV}(A, B)$ : 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 $\beta_j \not\prec \beta_i$
3. $R \leq R'$
Correction of $\mathcal{CW}(A, B):$ proof

For $1 \leq i \leq n$, we write $\alpha_i = \text{name}(S_i)$.

We have $\{A, B\} \cap \text{Princ}(\alpha_i) \neq \{A, B\}$.

**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 $\beta_j \not< \beta_i$
3. $R \leq R'$
Correction of $C\mathcal{W}(A, B)$: 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 $\beta_j \not< \beta_i$.
- $\{\gamma_i\}_{1 \leq i \leq k}$: 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 $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$ and $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not< \beta_j$. 
Correction of $\mathcal{CW}(A, B)$: 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 $\beta_j \not\prec \beta_i$.

$\{\gamma_i\}_{1 \leq i \leq k}$: 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 $C\mathcal{W}(A, B):$ 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 $R_1 : M \xrightarrow{\alpha} \beta N$, then we have $R_2 : M \xrightarrow{\beta} \alpha N$ and $R_1 \sim R_2$. 
Correction of $\mathcal{CW}(A, B)$: 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 $R_1 : M \Rightarrow \Rightarrow N$, then we have $R_2 : M \Rightarrow \Rightarrow N$ and $R_1 \sim R_2$. 
Correction of $\mathcal{CW}(A, B)$ : proof

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

$\beta_1 \downarrow$ $\beta_2 \downarrow$ $\beta_3 \downarrow$ $\cdots$ $\beta_m \downarrow$

$R \xrightarrow{\beta}$ $R' \xrightarrow{\beta_3}$ $R'/R$

$\{\eta_i\}_{1 \leq i \leq p}$ : 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 $\eta_i \not\preceq \theta_j$ and $\theta_j \not\preceq \eta_i$. 
Correction of $\mathcal{C}\mathcal{W}(A, B)$: proof

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

$M$ \quad $N'$

$\beta_1$ \quad $\beta_2$ \quad $\beta_3$ \quad $\beta_4$ \quad $\beta_5$

$R$ \quad $R'/R$

$\gamma_1$ \quad $\gamma_k$

$\delta_1$ \quad $\delta_2$

$\delta_{k'}$

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

$\theta_i \leq i \leq p'$ : 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 $CW(A, B)$ : 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 $CW(A, B)$: proof

1. $\{\eta_i\}_{1 \leq i \leq p}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{A\}$.
2. $\{\theta_i\}_{1 \leq i \leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \text{Princ}(\delta_i) = \{B\}$.
3. For every $i, j$, we have $\eta_i \not< \theta_j$ and $\theta_j \not< \eta_i$. 
Correction of $CW(A, B)$ : 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
Correction of $\mathcal{CWW}(A, B)$: proof
λ-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].
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: 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. 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. 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. 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. Simple proofs for safety properties.