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. λ-calculus, principals and independence
3. λ-calculus and the Chinese Wall
4. Future works

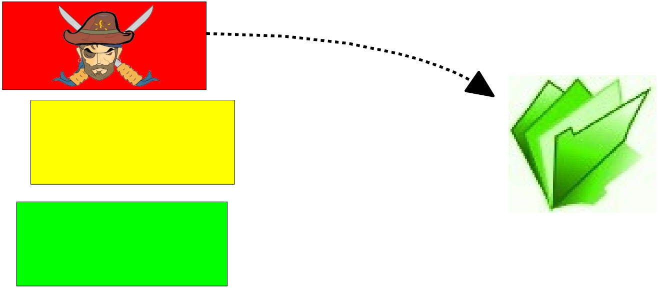
Motivations

Security and Programming languages

- Restricting rights of downloaded programs is not sufficient...

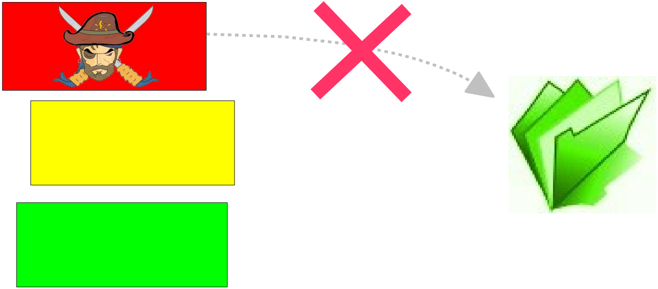
Security and Programming languages

- Restricting rights of downloaded programs is not sufficient...



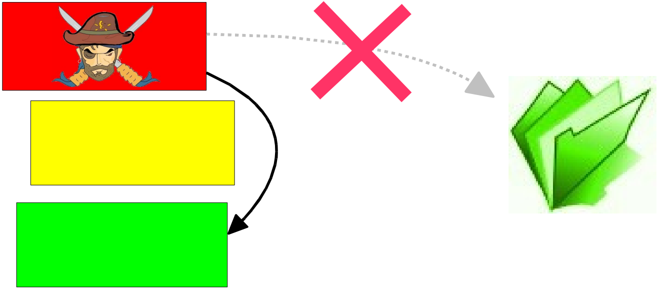
Security and Programming languages

- Restricting rights of downloaded programs is not sufficient...



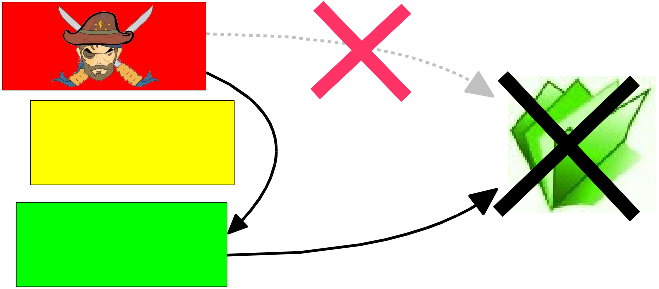
Security and Programming languages

- Restricting rights of downloaded programs is not sufficient...



Security and Programming languages

- Restricting rights of downloaded programs is not sufficient...
- ... since attackers can borrow privileges from local programs [Hardy].

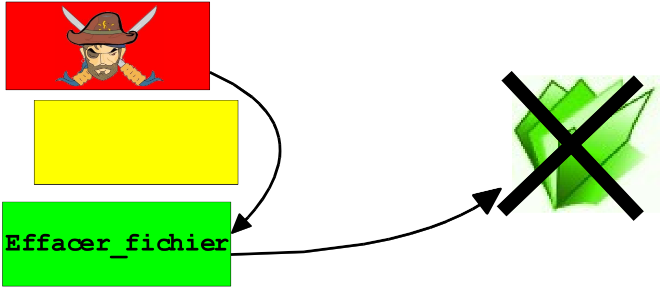


First approach : stack inspection

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

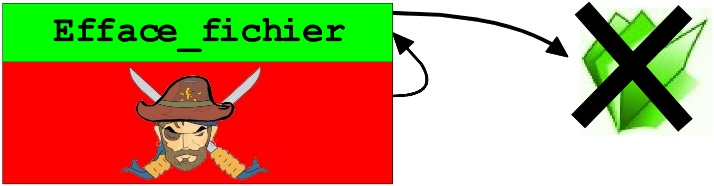
First approach : stack inspection

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



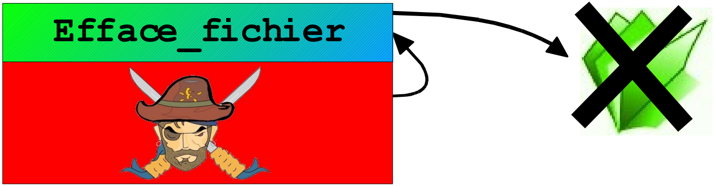
First approach : stack inspection

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



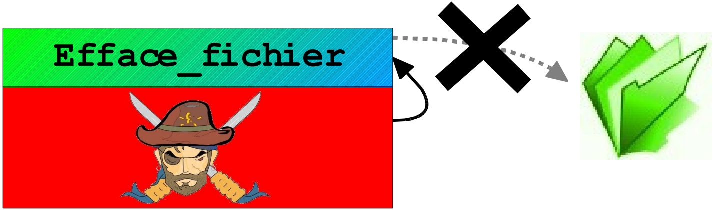
First approach : stack inspection

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



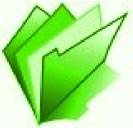
First approach : stack inspection

- 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

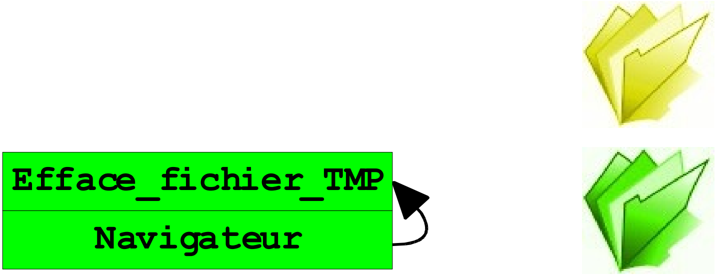
- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem : there remains (indirect) ways of acting outside function calls [Fournet-Gordon].



Navigateur

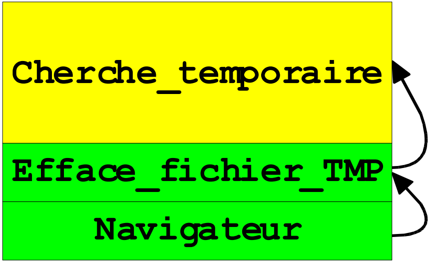
First approach : stack inspection

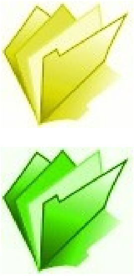
- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem : there remains (indirect) ways of acting outside function calls [Fournet-Gordon].



First approach : stack inspection

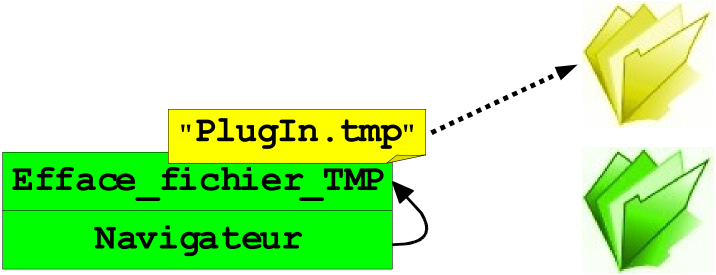
- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem : there remains (indirect) ways of acting outside function calls [Fournet-Gordon].





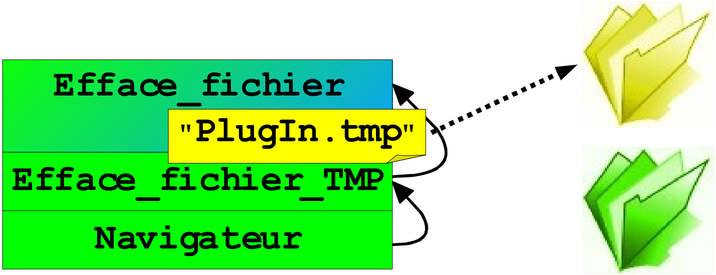
First approach : stack inspection

- 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]`.



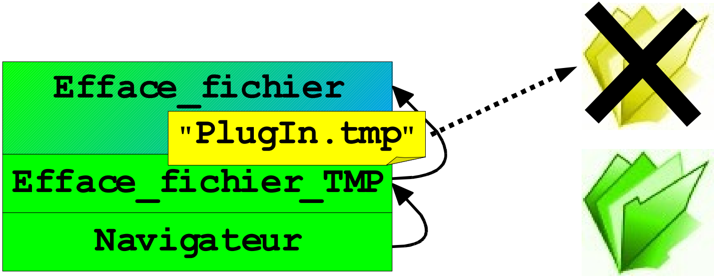
First approach : stack inspection

- 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].



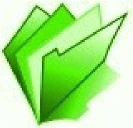
First approach : stack inspection

- 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].



First approach : stack inspection

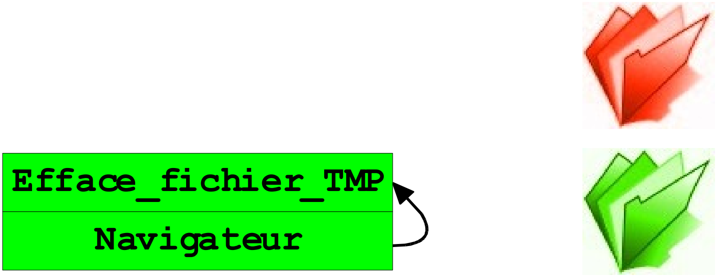
- Used in Java and C#.
- Before executing a sensitive action, one inspects the chain of function calls leading to that action.
- Problem : there remains (indirect) ways of acting outside function calls `[Fournet-Gordon]`.



Navigateur

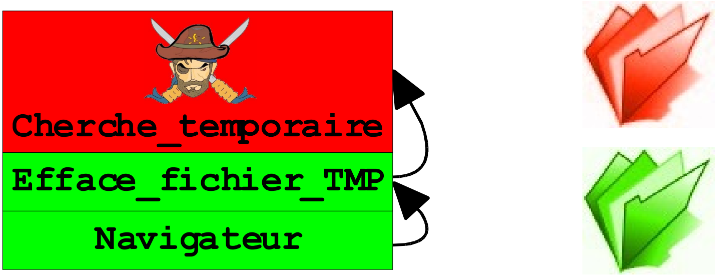
First approach : stack inspection

- 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].



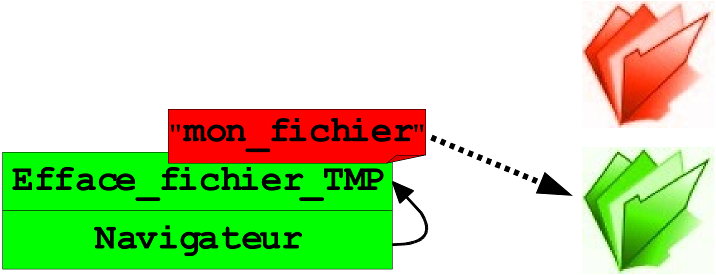
First approach : stack inspection

- 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].



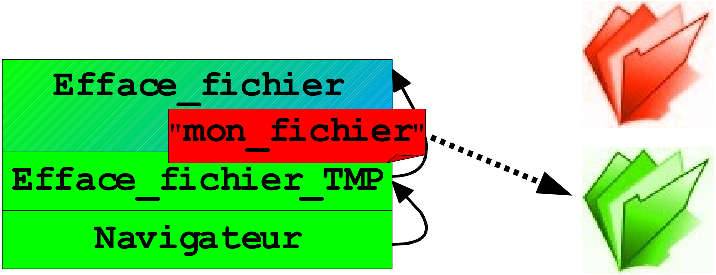
First approach : stack inspection

- 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].



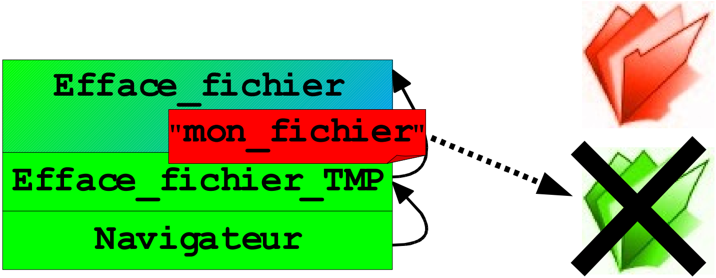
First approach : stack inspection

- 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].

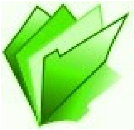


Second approach : Information Flow

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

Second approach : Information Flow

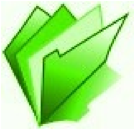
- Data are classified in several categories and their propagation is tracked during program execution.

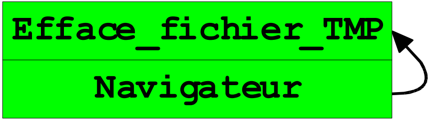


Navigateur

Second approach : Information Flow

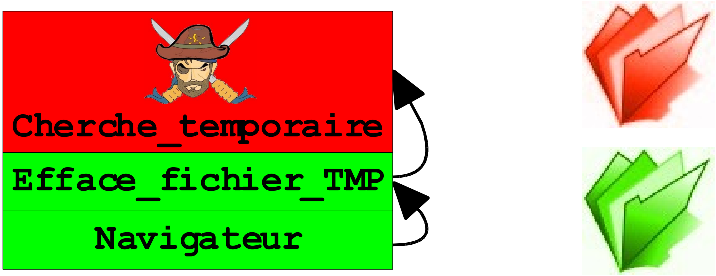
- Data are classified in several categories and their propagation is tracked during program execution.





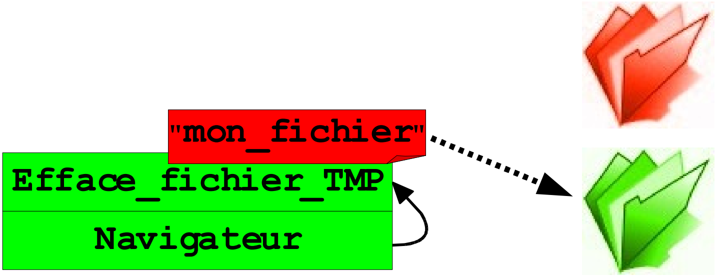
Second approach : Information Flow

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



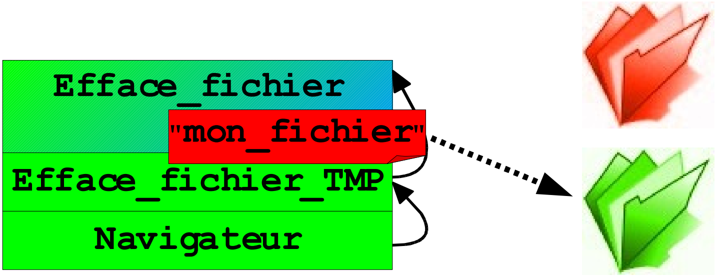
Second approach : Information Flow

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



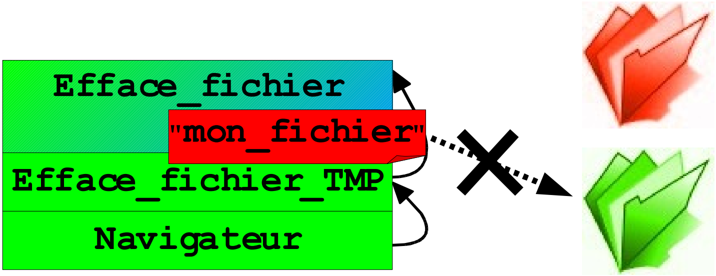
Second approach : Information Flow

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



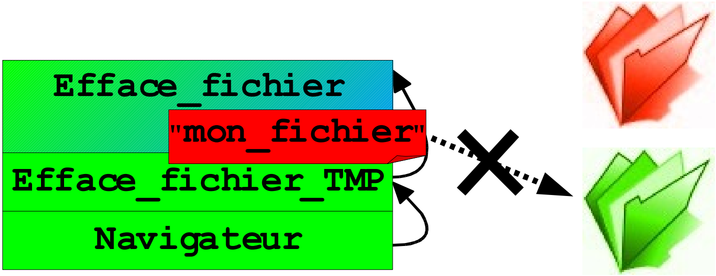
Second approach : Information Flow

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



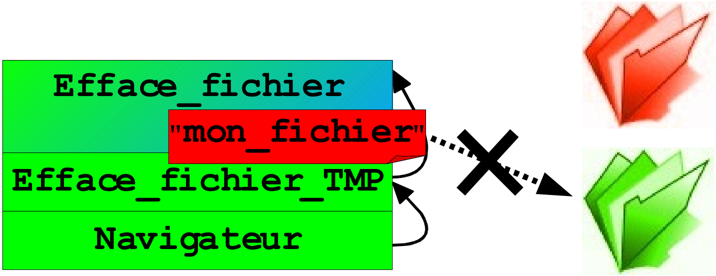
Second approach : Information Flow

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



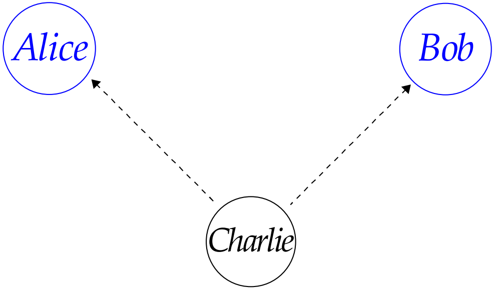
Second approach : Information Flow

- 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

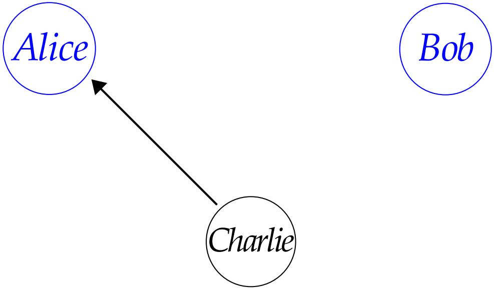
- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract ; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.



- Charlie may interact with Alice and Bob.

Third approach : the Chinese Wall

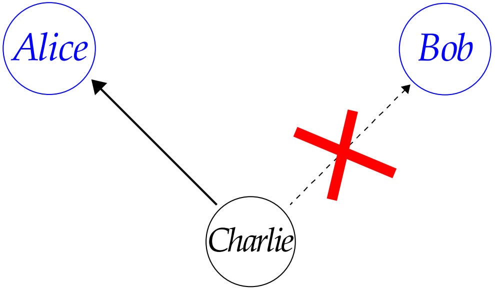
- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract ; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.



- Charlie may interact with Alice and Bob.
- But as soon as Charlie interacts with Alice...

Third approach : the Chinese Wall

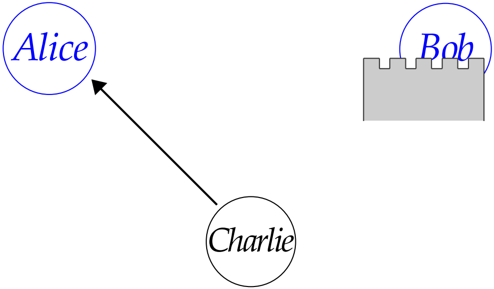
- Conflicts of interest in « economy » [Brewer-Nash].
- Alice and Bob compete for a contract ; Charlie is the buyer.
- Alice and Bob fix the price of the contract.
- Charlie wants to negotiate the price.



- 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

Safety policy	*Safety property*
Stack Inspection	-
Flow Information	Non interference
Chinese Wall	?

Objectives :

- define the Chinese Wall in the λ-calculus.
- examine the safety property of the Chinese Wall policy.

Summary

Safety policy	*Safety property*
Stack Inspection	-
Flow Information	Non interference
Chinese Wall	**?**

Objectives :

- define the Chinese Wall in the λ-calculus.
- examine the safety property of the Chinese Wall policy.

λ-calculus, principals and independence

λ_n-calculus : a λ-calculus with principals

- Alice, Bob, Charlie are principals.

$$A, B, \ldots$$

- Terms of λ_n-calculus :

$$\begin{array}{rcll} M,\ N & ::= & x & \textit{Variable} \\ & \mid & (\lambda x.M)^A & \textit{Abstraction} \\ & \mid & (MN)^A & \textit{Application} \end{array}$$

- Values :

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

- **Remark** : principals differ from labels in the labelled λ-calculus.

λ_n-calculus : a λ-calculus with principals

- Alice, Bob, Charlie are principals.

$$A, B, \ldots$$

- Terms of λ_n-calculus :

$$\begin{array}{rcll} M,\ N & ::= & x & \textit{Variable} \\ & | & (\lambda x.M)^A & \textit{Abstraction} \\ & | & (MN)^A & \textit{Application} \end{array}$$

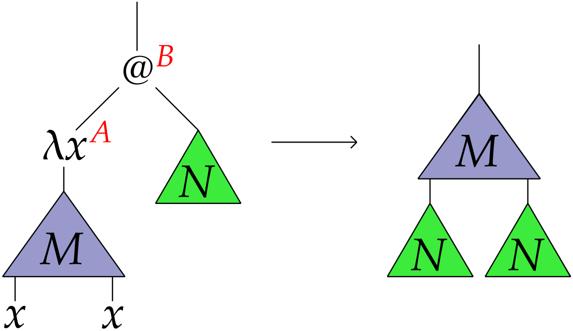
- Values :

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

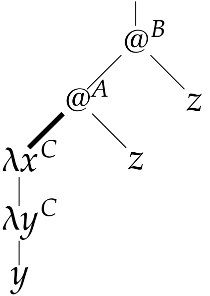
- **Remark** : principals differ from labels in the labelled λ-calculus.

Reduction in λ_n-calculus

$$(\beta) \quad ((\lambda x.M)^{A} N)^{B} \to M\{x \backslash N\}$$

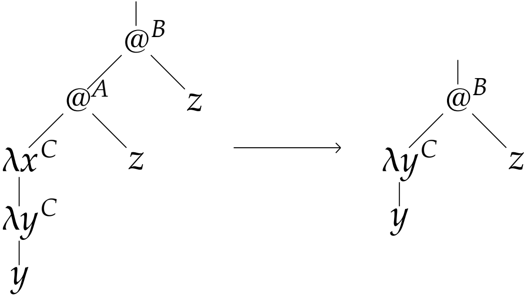


An example of reduction in the λ_n-calculus



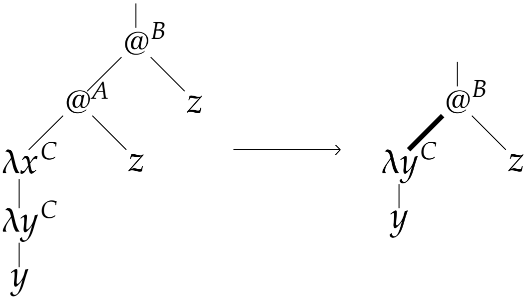
$$(((\lambda x.(\lambda y.y)^C)^C z)^A z)^B$$

An example of reduction in the λ_n-calculus



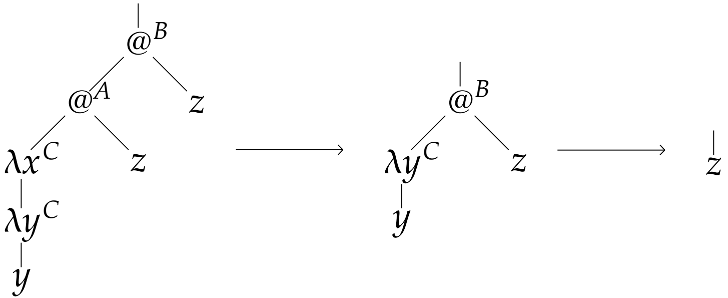
$$(((\lambda x.(\lambda y.y)^C)^C z)^A z)^B \quad \rightarrow \quad ((\lambda y.y)^C z)^B$$

An example of reduction in the λ_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 λ_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 λ_n-calculus

- Confluence
- Finite Developments
- Standardisation

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 $M \overset{\neg A}{\twoheadrightarrow} M'$ if every reduction step ignores A.

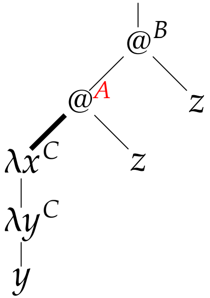
Reduction ignoring a principal

Definition

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

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

Example :



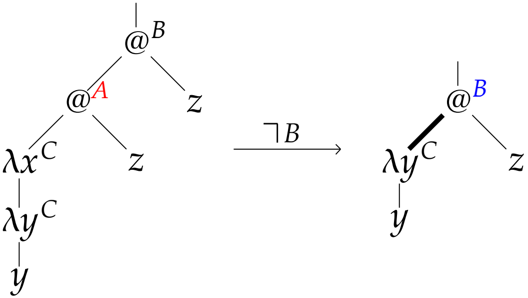
Reduction ignoring a principal

Definition

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

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

Example :



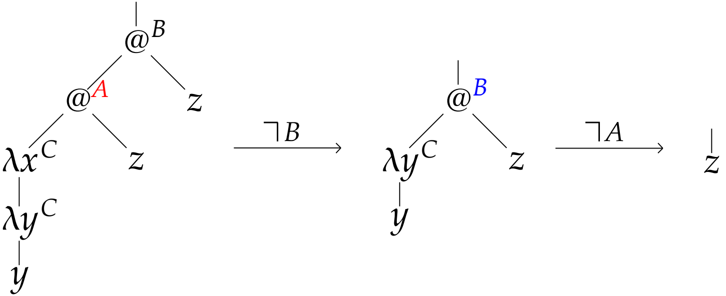
Reduction ignoring a principal

Definition

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

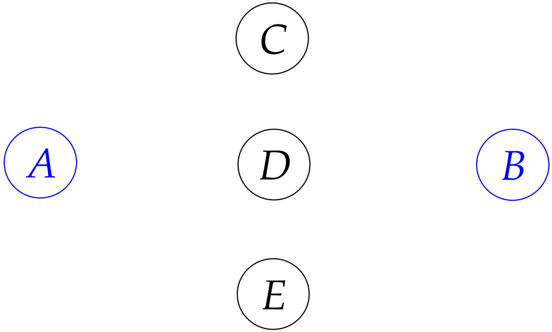
- Also written $M \xrightarrow{\neg A} M'$.
- We write $M \xrightarrow{\neg A}\!\!\!\!\rightarrow M'$ if every reduction step ignores A.

Example :



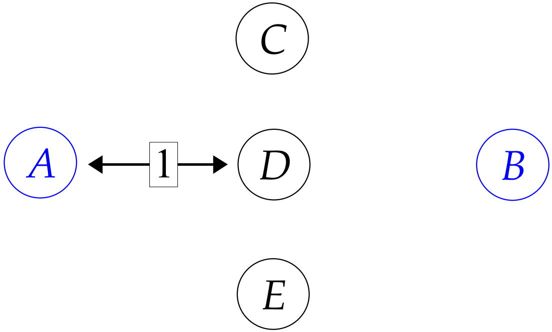
Independence

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



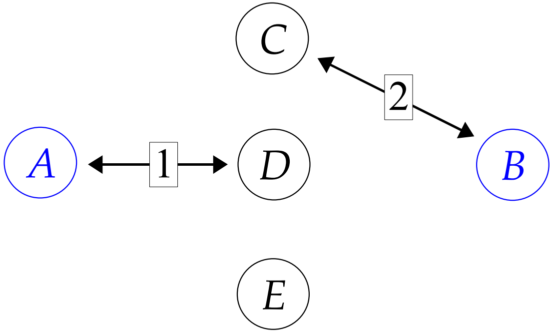
Independence

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



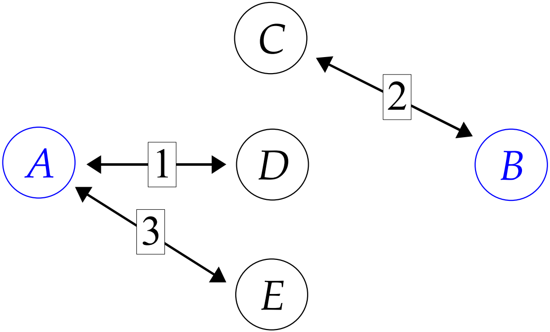
Independence

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



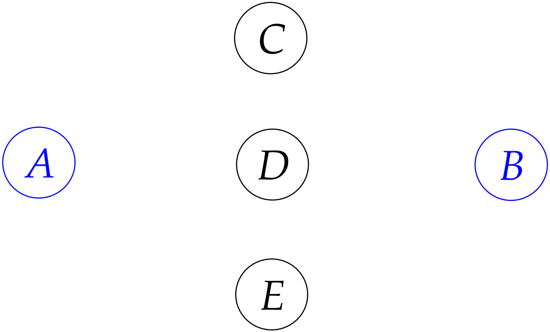
Independence

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



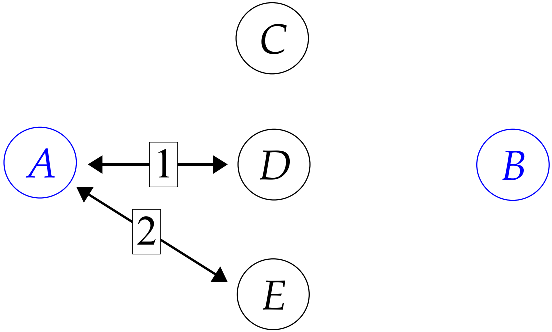
Independence

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



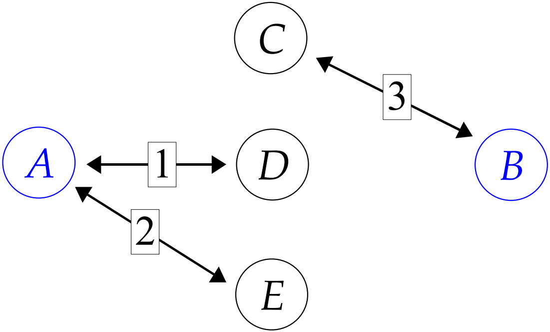
Independence

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



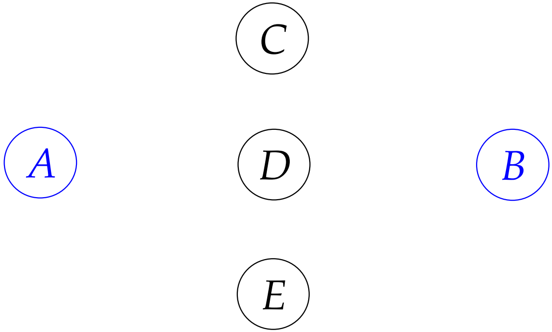
Independence

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



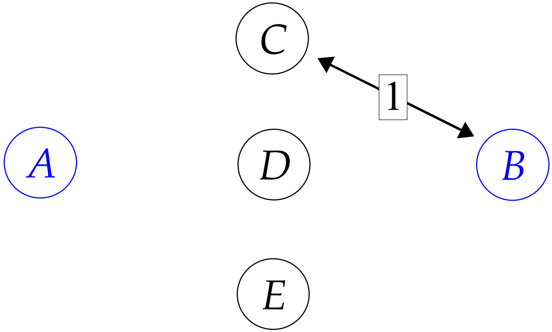
Independence

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



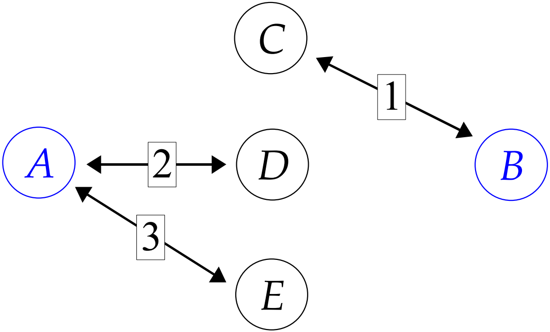
Independence

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



Independence

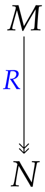
- Actions of A and B are independent if they commute.



Independence

Definition (Independence)

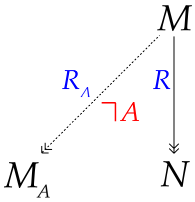
The reduction $R : M \twoheadrightarrow N$ is independent of the interaction between A and B iff there exists $R_A : M \xrightarrow{\neg A}\!\!\!\twoheadrightarrow M_A$ and $R_B : M \xrightarrow{\neg B}\!\!\!\twoheadrightarrow M_B$ such that $R \leq R'$ (i.e. R/R' is empty) with $R' = R_A;(R_B/R_A) =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



Independence

Definition (Independence)

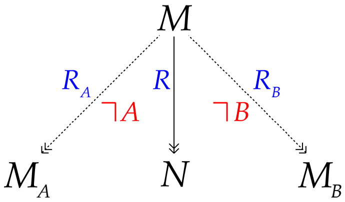
The reduction $R : M \twoheadrightarrow N$ is independent of the interaction between A and B iff there exists $R_A : M \xrightarrow{\neg A}\!\!\!\!\twoheadrightarrow M_A$ and $R_B : M \xrightarrow{\neg B}\!\!\!\!\twoheadrightarrow M_B$ such that $R \leq R'$ (i.e. R/R' is empty) with $R' = R_A;(R_B/R_A)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



Independence

Definition (Independence)

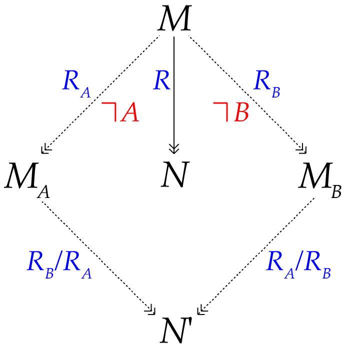
The reduction $R : M \twoheadrightarrow N$ is independent of the interaction between A and B iff there exists $R_A : M \xrightarrow{\neg A}\!\!\!\!\twoheadrightarrow M_A$ and $R_B : M \xrightarrow{\neg B}\!\!\!\!\twoheadrightarrow M_B$ such that $R \leq R'$ (i.e. R/R' is empty) with $R' = R_A;(R_B/R_A)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



Independence

Definition (Independence)

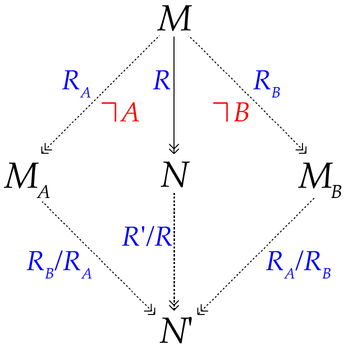
The reduction $R : M \twoheadrightarrow N$ *is* *independent* *of the interaction between* A *and* B *iff there exists* $R_A : M \xrightarrow{\neg A}\!\!\!\!\rightarrow M_A$ *and* $R_B : M \xrightarrow{\neg B}\!\!\!\!\rightarrow M_B$ *such that* $R \leq R'$ *(i.e.* R/R' *is empty) with* $R' = R_A; (R_B/R_A)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



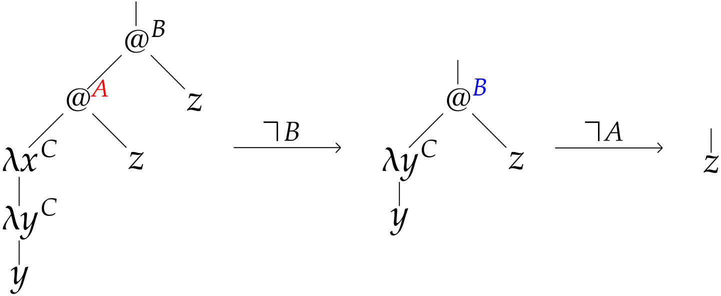
Independence

Definition (Independence)

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

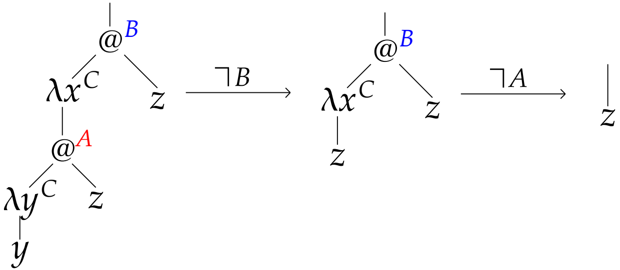


Independence : example 1/2



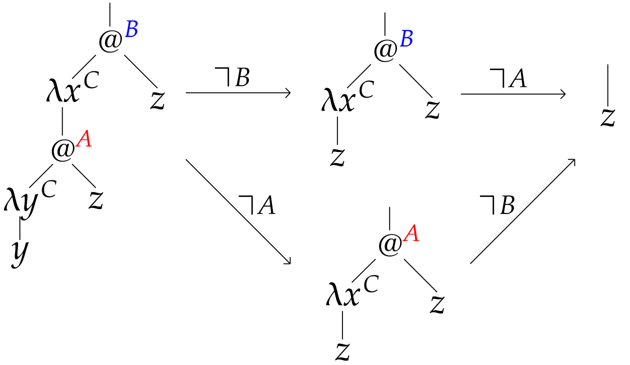
This reduction is not independent of the interaction between A and B.

Independence : example 2/2



This reduction is independent of the interaction between A and B.

Independence : example 2/2



This reduction is independent of the interaction between A and B.

The λ_n-calculus : summary

- A λ-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the λ_n-calculus ?
  - This policy relies on history.
  - We use the labelled λ-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 λ_n-calculus : summary

- A λ-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the λ_n-calculus ?
  - This policy relies on history.
  - We use the labelled λ-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 λ_n-calculus : summary

- A λ-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the λ_n-calculus ?
  - This policy relies on history.
  - We use the labelled λ-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 λ_n-calculus : summary

- A λ-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the λ_n-calculus ?
  - This policy relies on history.
  - We use the labelled λ-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 λ_n-calculus : summary

- A λ-calculus with principals.
- A safety property : independence.
- How to express the Chinese Wall policy in the λ_n-calculus ?
  - This policy relies on history.
  - We use the labelled λ-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.

λ-calculus

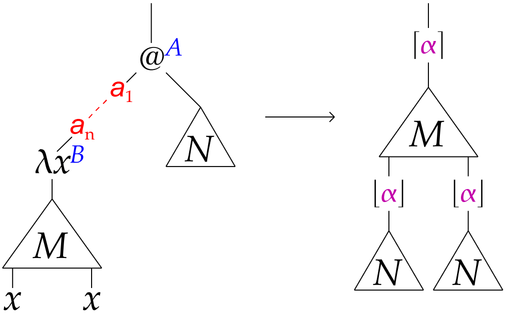
and

the Chinese Wall

The labelled λ_n-calculus

Terms	$M,\ N$	$::=$	x
		$\mid$	$(\lambda x.N)^A$
		$\mid$	$(MN)^A$
		$\mid$	$a : M$
Atomic labels	$a,\ b$	$::=$	$\lceil \alpha \rceil \mid \lfloor \alpha \rfloor$
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

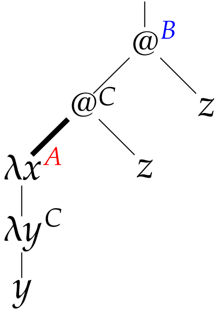


$$(\beta) \quad R = (a_1 : \ldots : a_n : (\lambda x.M)^B N)^A \rightarrow \lceil \alpha \rceil : M\{x \backslash \lfloor \alpha \rfloor : N\}$$

$$\alpha = A a_1 \ldots a_n B$$

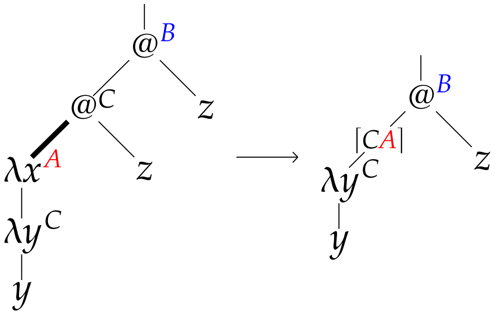
The redex name is $\mathtt{name}(R) = \alpha$.

Labelled reduction : an example



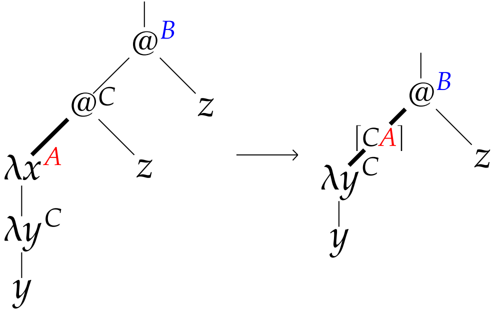
$$(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B$$

Labelled reduction : an example



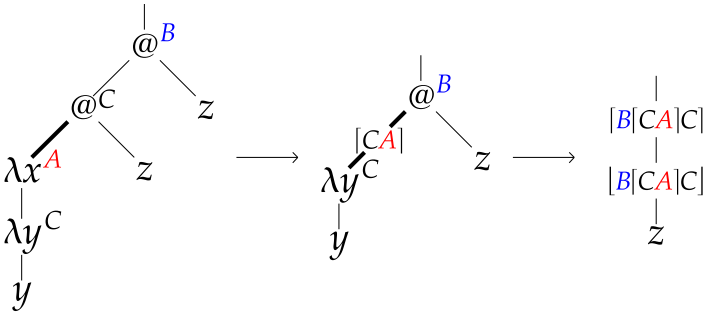
$$(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B \to (\lceil CA \rceil : (\lambda y.y)^C \; z)^B$$

Labelled reduction : an example



$$(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B \to ([\lceil CA \rceil : (\lambda y.y)^C \; z)^B$$

Labelled reduction : an example



$$
\begin{aligned}
(((\lambda x.(\lambda y.y)^C)^A z)^C z)^B &\to (\lceil CA \rceil : (\lambda y.y)^C\ z)^B \\
&\to \lceil B \lceil CA \rceil C \rceil : \lfloor B \lceil CA \rceil C \rfloor : z
\end{aligned}
$$

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 :

$$\begin{aligned} \mathtt{Princ}(Aa_1 \ldots a_n B) &= \{A, B\} \cup_{1 \le i \le n} \mathtt{Princ}(a_i) \\ \mathtt{Princ}(\lceil \alpha \rceil) &= \mathtt{Princ}(\lfloor \alpha \rfloor) = \mathtt{Princ}(\alpha) \end{aligned}$$

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 :

$$\begin{aligned}\texttt{Princ}(Aa_1 \ldots a_n B) &= \{A, B\} \cup_{1 \leq i \leq n} \texttt{Princ}(a_i) \\ \texttt{Princ}(\lceil \alpha \rceil) &= \texttt{Princ}(\lfloor \alpha \rfloor) = \texttt{Princ}(\alpha)\end{aligned}$$

  - Example : $\texttt{Princ}(A\lceil B\lfloor AC\rfloor D\rceil E) = \{A, B, C, D, E\}$

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 :

$$\begin{aligned} \mathtt{Princ}(Aa_1 \dots a_n B) &= \{A, B\} \cup_{1 \leq i \leq n} \mathtt{Princ}(a_i) \\ \mathtt{Princ}(\lceil \alpha \rceil) &= \mathtt{Princ}(\lfloor \alpha \rfloor) = \mathtt{Princ}(\alpha) \end{aligned}$$

Definition (Separation)

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

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 :

$$\begin{aligned}\mathtt{Princ}(Aa_1 \ldots a_n B) &= \{A, B\} \cup_{1 \leq i \leq n} \mathtt{Princ}(a_i) \\ \mathtt{Princ}(\lceil \alpha \rceil) &= \mathtt{Princ}(\lfloor \alpha \rfloor) = \mathtt{Princ}(\alpha)\end{aligned}$$

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 $\{A, B\} \not\subseteq \mathtt{Princ}(a_i)$.

- Examples :
  - ⋆ $\lceil AC \rceil \lfloor C \lceil DE \rceil B \rfloor$ separates A et B.
  - ⋆ $\lceil DC \rceil \lfloor C \lceil AE \rceil B \rfloor$ does not separate A et B.

Independence and labels : separation

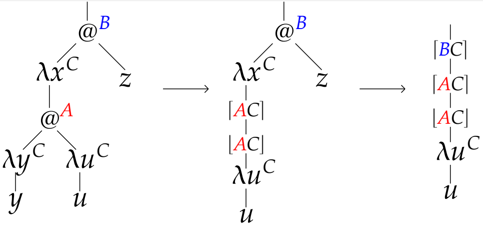
Theorem (Separation)

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

Independence and labels : separation

Theorem (Separation)

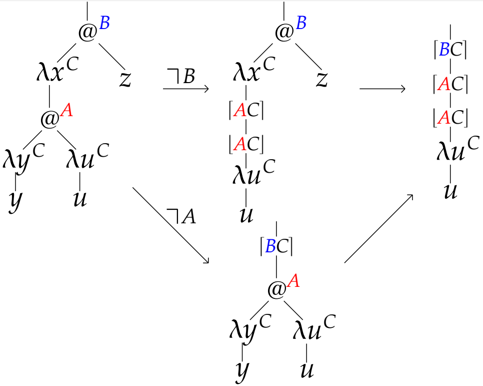
If M is an unlabelled term and if the reduction $M \twoheadrightarrow V$ is independent of the interaction between A and B, then $\tau(V)$ separates A and B.



Independence and labels : separation

Theorem (Separation)

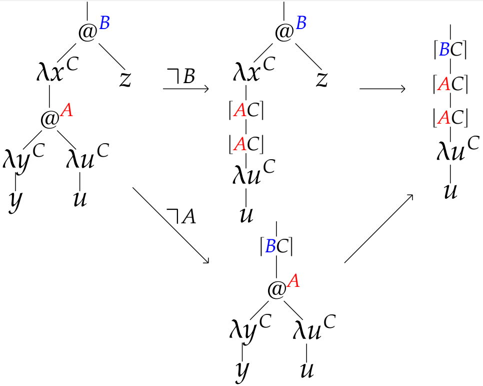
If M is an unlabelled term and if the reduction $M \twoheadrightarrow V$ is independent of the interaction between A and B, then $\tau(V)$ separates A and B.



Independence and labels : separation

Theorem (Separation)

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



The head sequence $\lceil BC \rceil \lceil AC \rceil \lfloor AC \rfloor$ separates A and B.

Independence and separation

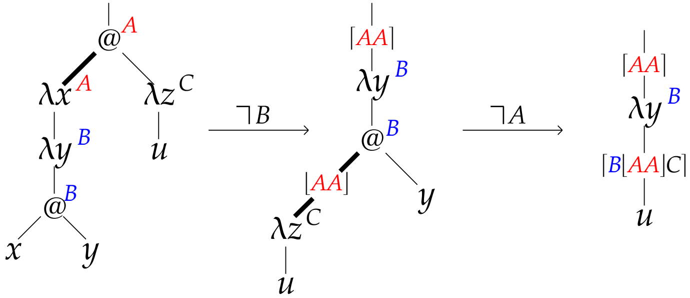
Theorem

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

Independence and separation

Theorem

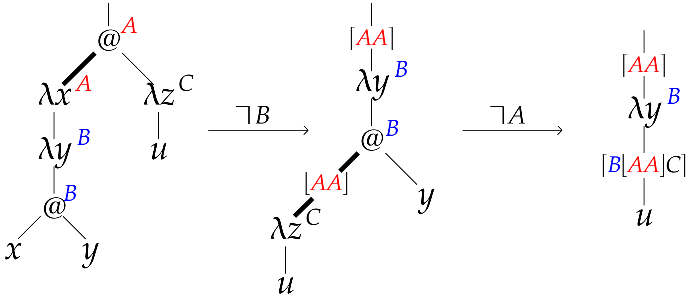
If $M \twoheadrightarrow V$ and if $\tau(V)$ separates A and B, then there is a reduction $\mathcal{R} : M \twoheadrightarrow W$ independent of the interaction between A and B.



Independence and separation

Theorem

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

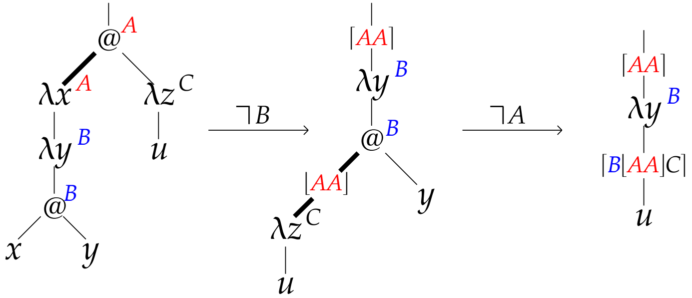


★ The label $\lceil AA \rceil$ separates A et B.

Independence and separation

Theorem

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

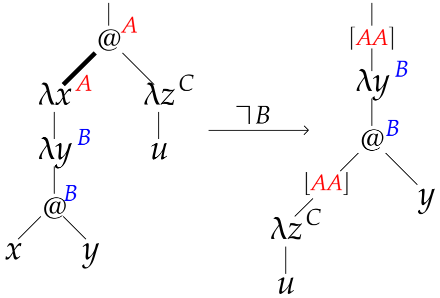


- The label $\lceil AA \rceil$ separates A et B.
- This reduction **is not** independent of the interaction between A and B.

Independence and separation

Theorem

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



- The label $\lceil AA \rceil$ separates A et B.
- This reduction **is** independent of the interaction between A and B.

Expressing Chinese Wall in the λ_n-calculus

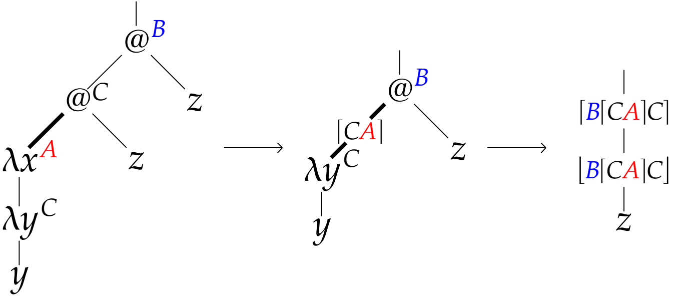
- The Chinese Wall between A and B is written $\mathcal{CW}(A, B)$.

Definition (Chinese Wall)

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

$$\{A, B\} \not\subseteq \texttt{Princ}(\texttt{name}(R))$$

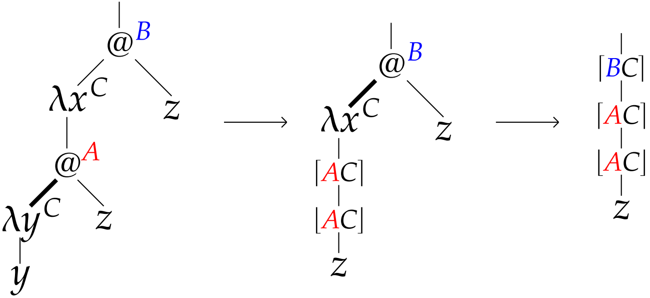
Chinese Wall in the λ_n-calculus : example 1/2



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

This reduction does not follow $\mathcal{CW}(A, B)$.

Chinese Wall in the λ_n-calculus : example 2/2



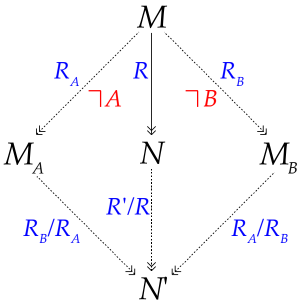
$$\texttt{Princ}(\texttt{name}(R_1)) = \{A, C\}$$
$$\texttt{Princ}(\texttt{name}(R_2)) = \{B, C\}$$

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

Correction of $\mathcal{CW}(A, B)$

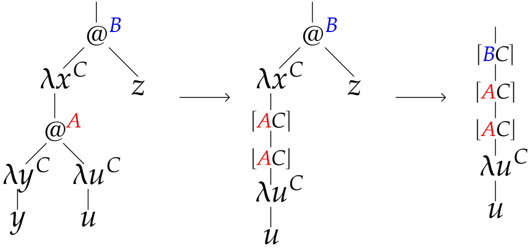
Theorem (Correction)

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



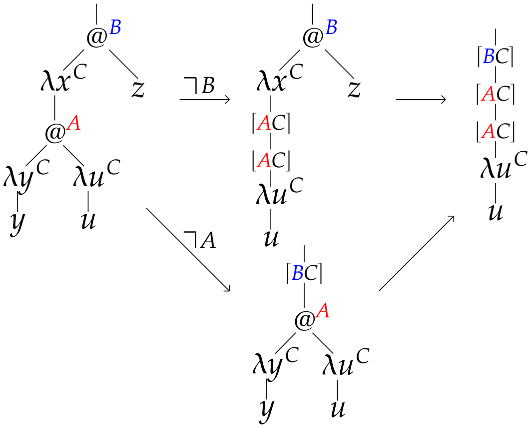
The Chinese Wall guarantees the independence.

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

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

- Sublabel of a compound label :

$$
\begin{aligned}
\alpha &\preceq \alpha \\
\alpha &\preceq Aa_1 \ldots a_n B \text{ si } \exists i \,.\, a_i = \lceil \beta \rceil \text{ and } \alpha \preceq \beta \\
\alpha &\preceq Aa_1 \ldots a_n B \text{ si } \exists i \,.\, a_i = \lfloor \beta \rfloor \text{ and } \alpha \preceq \beta
\end{aligned}
$$

- Example : $\alpha \preceq A\lceil\alpha\rceil\lfloor\gamma\rfloor B$

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

- Sublabel of a compound label :

$$
\begin{array}{lll}
\alpha & \preceq & \alpha \\
\alpha & \preceq & Aa_1 \ldots a_n B \text{ si } \exists i \ . \ a_i = \lceil \beta \rceil \text{ and } \alpha \preceq \beta \\
\alpha & \preceq & Aa_1 \ldots a_n B \text{ si } \exists i \ . \ a_i = \lfloor \beta \rfloor \text{ and } \alpha \preceq \beta
\end{array}
$$

- Example : $\alpha \preceq A\lceil \alpha \rceil \lfloor \gamma \rfloor B$

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

$$M \xrightarrow{S_1} \xrightarrow{S_2} \underset{R}{\xrightarrow{S_3}} \text{- - - -} \xrightarrow{S_n} N$$

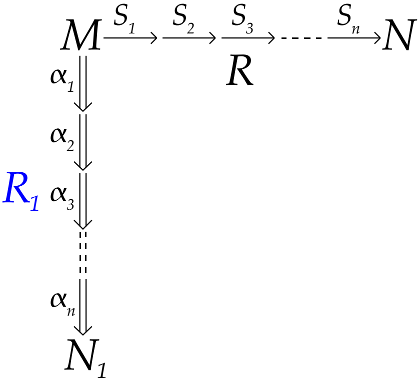
Correction of $\mathcal{CW}(A, B)$: proof

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

$$R$$

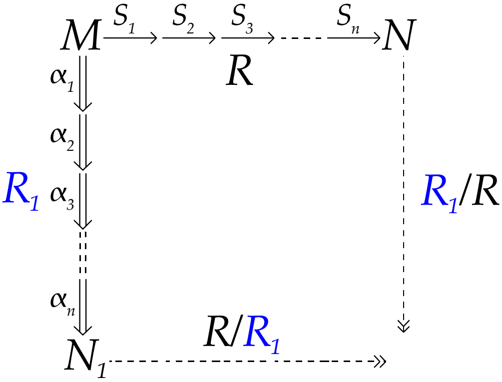
For $1 \leq i \leq n$, we write $\alpha_i = \texttt{name}(S_i)$.
We have $\{A, B\} \cap \texttt{Princ}(\alpha_i) \neq \{A, B\}$.

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



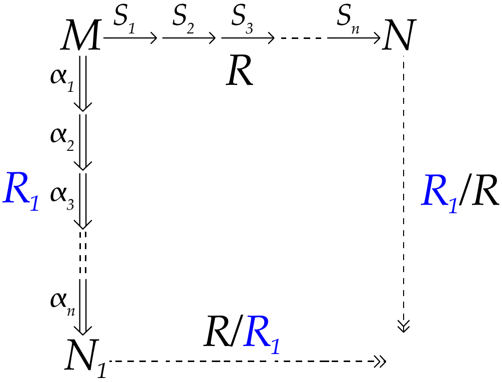
For $1 \leq i \leq n$, we write $\alpha_i = \mathtt{name}(S_i)$.
We have $\{A, B\} \cap \mathtt{Princ}(\alpha_i) \neq \{A, B\}$.

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



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

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

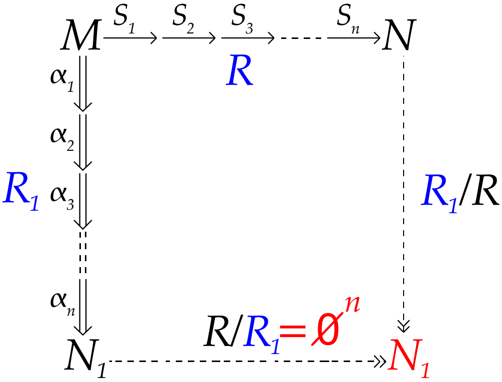


For $1 \leq i \leq n$, we write $\alpha_i = \mathtt{name}(S_i)$.
We have $\{A, B\} \cap \mathtt{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Completion)

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

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

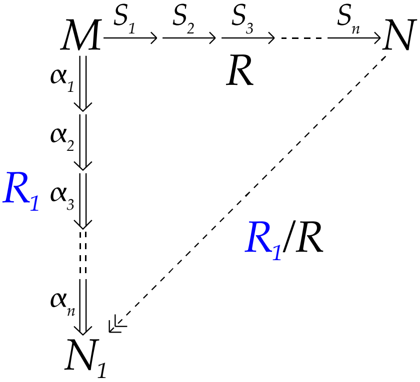


For $1 \leq i \leq n$, we write $\alpha_i = \texttt{name}(S_i)$.
We have $\{A, B\} \cap \texttt{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Completion)

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

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

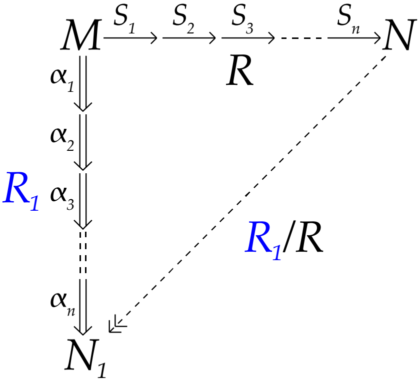


For $1 \leq i \leq n$, we write $\alpha_i = \texttt{name}(S_i)$.
We have $\{A, B\} \cap \texttt{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Completion)

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

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

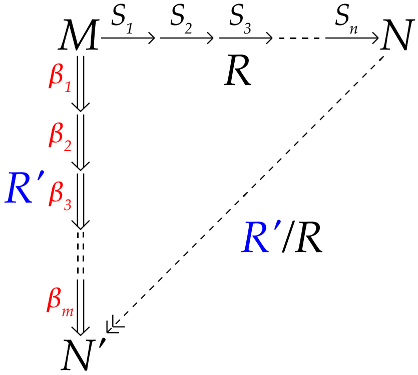


For $1 \leq i \leq n$, we write $\alpha_i = \mathtt{name}(S_i)$.
We have $\{A, B\} \cap \mathtt{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Reordering)

If $R : M \overset{\alpha_1}{\Longrightarrow} \dots \overset{\alpha_n}{\Longrightarrow} N$*, there is a reduction* $R' : M \overset{\beta_1}{\Longrightarrow} \dots \overset{\beta_m}{\Longrightarrow} N'$ *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 $\mathcal{CW}(A, B)$: proof

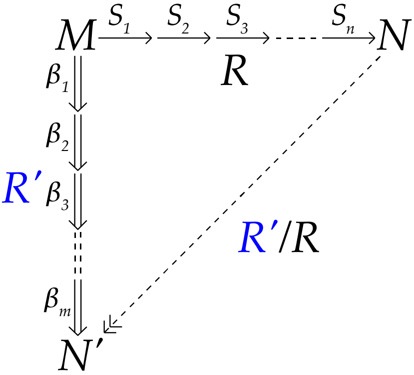


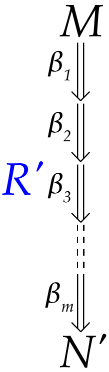
For $1 \leq i \leq n$, we write $\alpha_i = \mathtt{name}(S_i)$.
We have $\{A, B\} \cap \mathtt{Princ}(\alpha_i) \neq \{A, B\}$.

Lemma (Reordering)

If $R : M \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_n}{\Longrightarrow} N$, there is a reduction $R' : M \overset{\beta_1}{\Longrightarrow} \ldots \overset{\beta_m}{\Longrightarrow} N'$ 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) $R \leq R'$

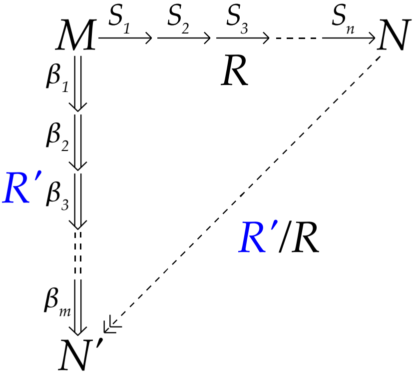
Correction of $\mathcal{CW}(A, B)$: proof

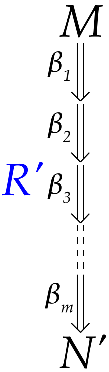




- If $i < j$, we have $\beta_j \not\prec \beta_i$.
- $\{\gamma_i\}_{1\leq i\leq k}$: elements of $\{\beta_i\}_{1\leq i\leq m}$ such that $\{A, B\} \cap \mathtt{Princ}(\beta_i) = \emptyset$.
- $\{\delta_i\}_{1\leq i\leq k'}$: elements of $\{\beta_i\}_{1\leq i\leq m}$ such that $\{A, B\} \cap \mathtt{Princ}(\beta_i)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

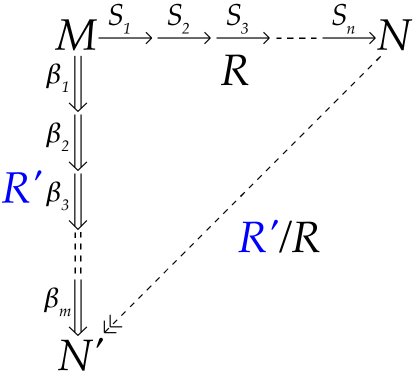
Correction of $\mathcal{CW}(A, B)$: proof

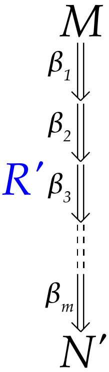




- If $i < j$, we have $\beta_j \not\prec \beta_i$.
- $\{\gamma_i\}_{1 \leq i \leq k}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \mathtt{Princ}(\beta_i) = \emptyset$.
- $\{\delta_i\}_{1 \leq i \leq k'}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \mathtt{Princ}(\beta_i)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

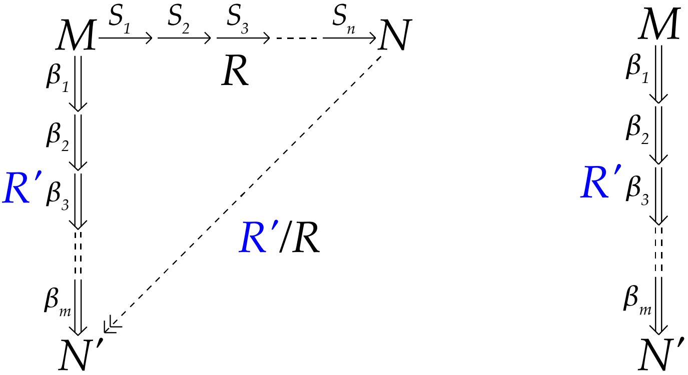
Correction of $\mathcal{CW}(A, B)$: proof





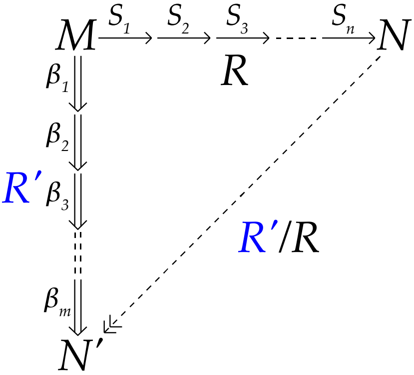
- If $i < j$, we have $\beta_j \not< \beta_i$.
- $\{\gamma_i\}_{1 \leq i \leq k}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \texttt{Princ}(\beta_i) = \emptyset$.
- $\{\delta_i\}_{1 \leq i \leq k'}$: elements of $\{\beta_i\}_{1 \leq i \leq m}$ such that $\{A, B\} \cap \texttt{Princ}(\beta_i)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

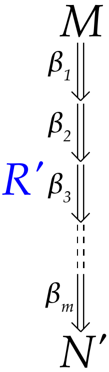
Correction of $\mathcal{CW}(A, B)$: proof



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



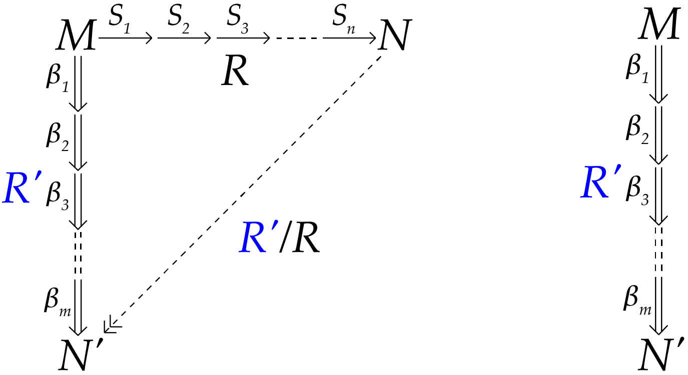


- If $i < j$ and $\beta_i \in \{\delta_i\}_{1 \le i \le k'}$ and $\beta_j \in \{\gamma_i\}_{1 \le i \le k}$, we have $\beta_i \not\prec \beta_j$ and $\beta_j \not\prec \beta_i$.

Lemma (Permutation)

If $\alpha \not\prec \beta$ and $\beta \not\prec \alpha$ and if $R_1 : M \overset{\alpha}{\Rightarrow}\overset{\beta}{\Rightarrow} N$, then we have $R_2 : M \overset{\beta}{\Rightarrow}\overset{\alpha}{\Rightarrow} N$ and $R_1 \sim R_2$.

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

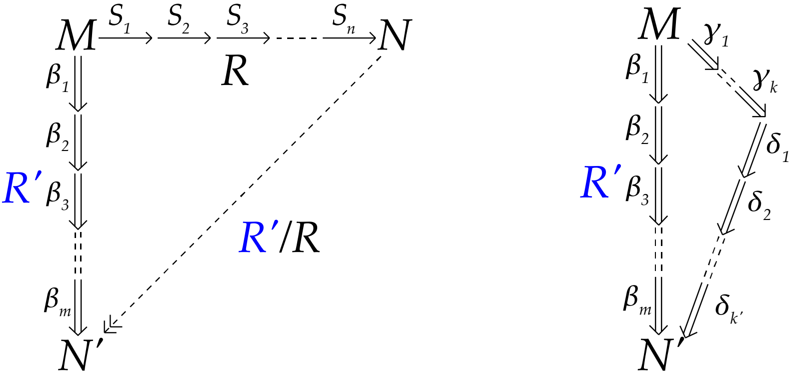


- If $i < j$ and $\beta_i \in \{\delta_i\}_{1 \leq i \leq k'}$ and $\beta_j \in \{\gamma_i\}_{1 \leq i \leq k}$, we have $\beta_i \not\prec \beta_j$ and $\beta_j \not\prec \beta_i$.

Lemma (Permutation)

If $\alpha \not\prec \beta$ and $\beta \not\prec \alpha$ and if $R_1 : M \overset{\alpha}{\Rightarrow}\overset{\beta}{\Rightarrow} N$, then we have $R_2 : M \overset{\beta}{\Rightarrow}\overset{\alpha}{\Rightarrow} N$ and $R_1 \sim R_2$.

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

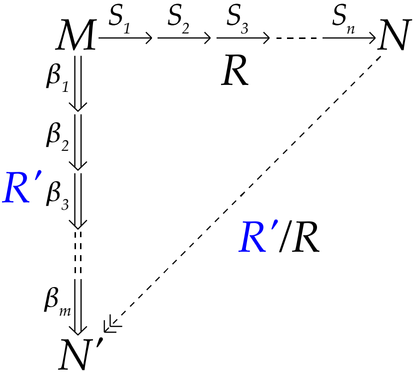


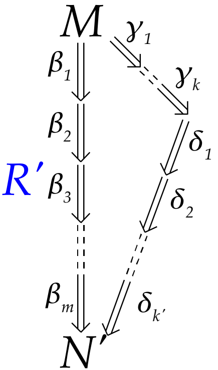
- If $i < j$ et $\beta_i \in \{\delta_i\}_{1\leq i\leq k'}$ and $\beta_j \in \{\gamma_i\}_{1\leq i\leq k}$, we have $\beta_i \not< \beta_j$ et $\beta_j \not< \beta_i$.

Lemma (Permutation)

If $\alpha \not< \beta$ and $\beta \not< \alpha$ and if $R_1 : M \overset{\alpha}{\Rightarrow}\overset{\beta}{\Rightarrow} N$, then we have $R_2 : M \overset{\beta}{\Rightarrow}\overset{\alpha}{\Rightarrow} N$ and $R_1 \sim R_2$.

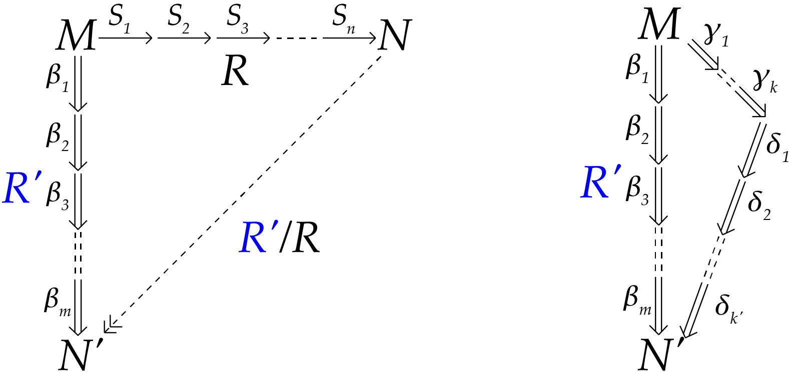
Correction of $\mathcal{CW}(A, B)$: proof





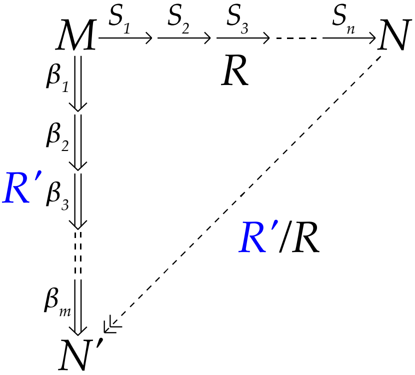
- $\{\eta_i\}_{1 \leq i \leq p}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \mathtt{Princ}(\delta_i) = \{A\}$.
- $\{\theta_i\}_{1 \leq i \leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \mathtt{Princ}(\delta_i) = \{B\}$.
- For every i, j, we have $\eta_i \not< \theta_j$ and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

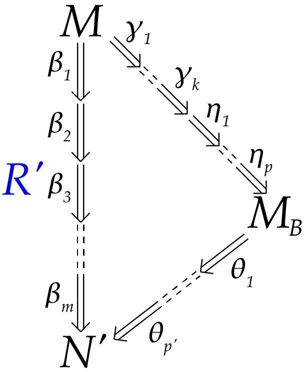
Correction of $\mathcal{CW}(A, B)$: proof



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

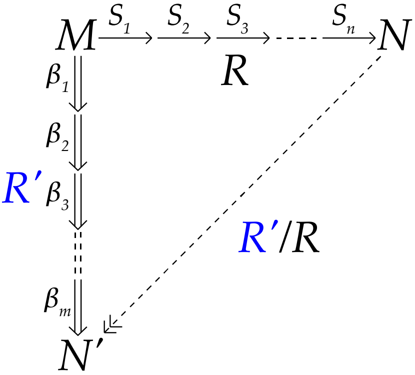
Correction of $\mathcal{CW}(A, B)$: proof

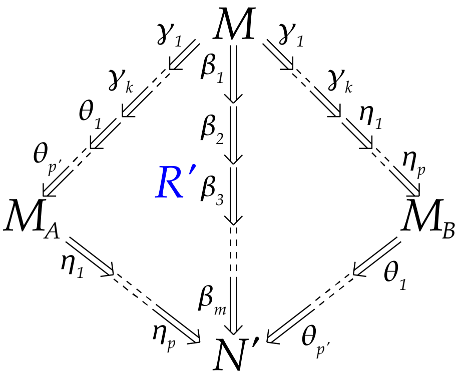




- $\{\eta_i\}_{1 \leq i \leq p}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \mathtt{Princ}(\delta_i) = \{A\}$.
- $\{\theta_i\}_{1 \leq i \leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \mathtt{Princ}(\delta_i) = \{B\}$.
- For every i, j, we have $\eta_i \not\prec \theta_j$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

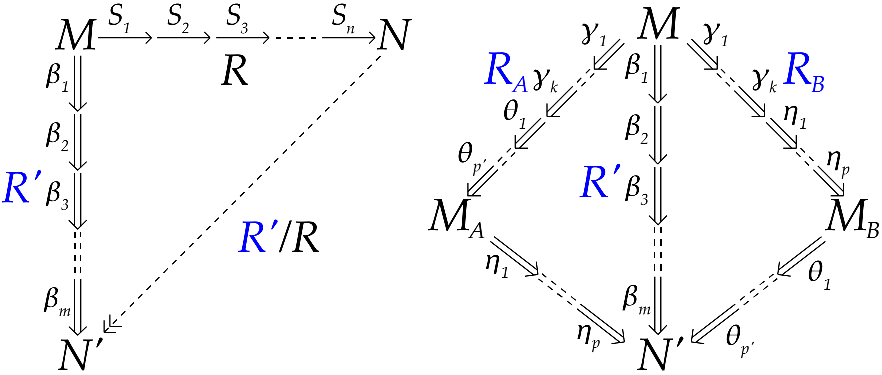
Correction of $\mathcal{CW}(A,B)$: proof





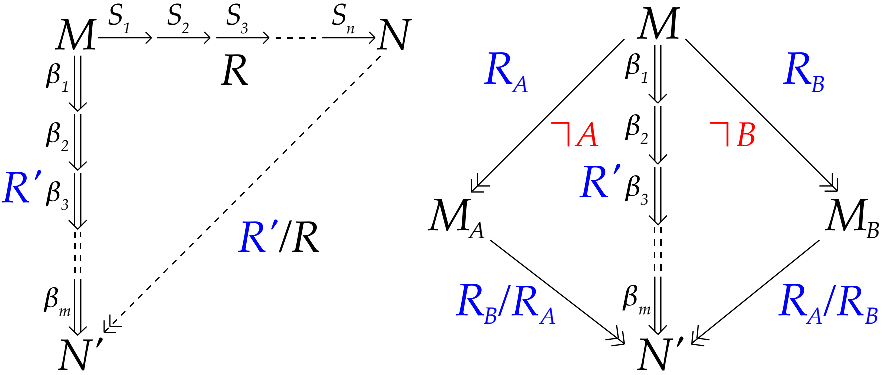
- $\{\eta_i\}_{1\leq i\leq p}$: elements of $\{\delta_i\}_i$ such that $\{A,B\}\cap \texttt{Princ}(\delta_i)=\{A\}$.
- $\{\theta_i\}_{1\leq i\leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A,B\}\cap \texttt{Princ}(\delta_i)=\{B\}$.
- For every i,j, we have $\eta_i \not< \theta_j$ and $\theta_j \not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

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

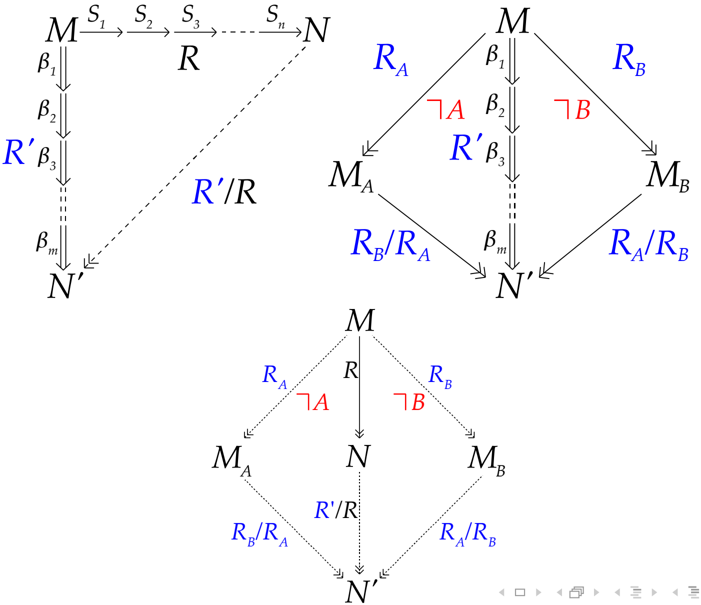


- $\{\eta_i\}_{1 \leq i \leq p}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \texttt{Princ}(\delta_i) = \{A\}$.
- $\{\theta_i\}_{1 \leq i \leq p'}$: elements of $\{\delta_i\}_i$ such that $\{A, B\} \cap \texttt{Princ}(\delta_i) = \{B\}$.
- For every i, j, we have $\eta_i \not\prec \theta_j$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

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



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



λ-calculus and Chinese Wall : summary

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

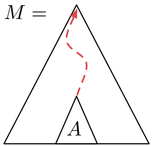
Safety policy	*Safety property*
Stack inspection	-
Information flow	Non interference
Chinese Wall	Independence

Future works

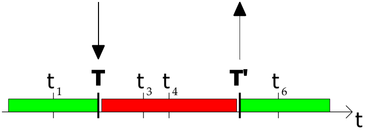
Objectives

1. Static information flow in the λ-calculus
   - labelled λ-calculus and DCC [Riecke], FlowCaml as [Simonet, Pottier], DCC+ [Abadi], etc
2. Reduction strategies
   - call-by-value λ-calculus
   - weak λ-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

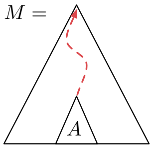


- Non interference : the labels of the λ-calculus express functional interference.
- In the λ-calculus with references, labels have to also capture interference with memory.
  - A memory cell interferes within some time interval.

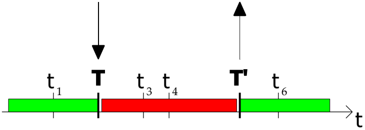


  - We can use irreversibility of contexts in the labelled λ-calculus [Blanc].

Conclusion : non interference

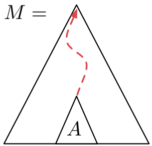


- Non interference : the labels of the λ-calculus express functional interference.
- In the λ-calculus with references, labels have to also capture interference with memory.
  - A memory cell interferes within some time interval.

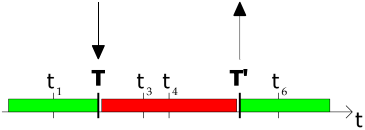


  - We can use irreversibility of contexts in the labelled λ-calculus [Blanc].

Conclusion : non interference

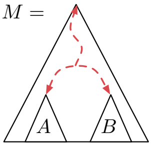


- Non interference : the labels of the λ-calculus express functional interference.
- In the λ-calculus with references, labels have to also capture interference with memory.
  - A memory cell interferes within some time interval.



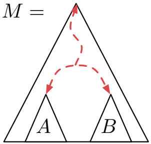
  - We can use irreversibility of contexts in the labelled λ-calculus [Blanc].

Conclusion : independence



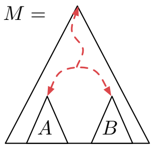
1. Created principals and extended independence.
2. Link between non-interference and independence : express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that kCFA is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

Conclusion : independence



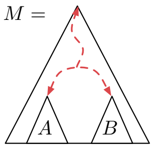
1. Created principals and extended independence.
2. Link between non-interference and independence : express these properties within a common framework.
3. Dynamic labels are a good starting point for an analysis mixing static and dynamic checks.
4. David Van Horn and Harry Mairson showed that kCFA is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

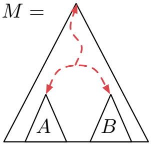
Conclusion : independence



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 kCFA 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 kCFA is NP as soon as $k > 0$. [ICFP 07].
5. Simple proofs for safety properties.