A language over money transfer

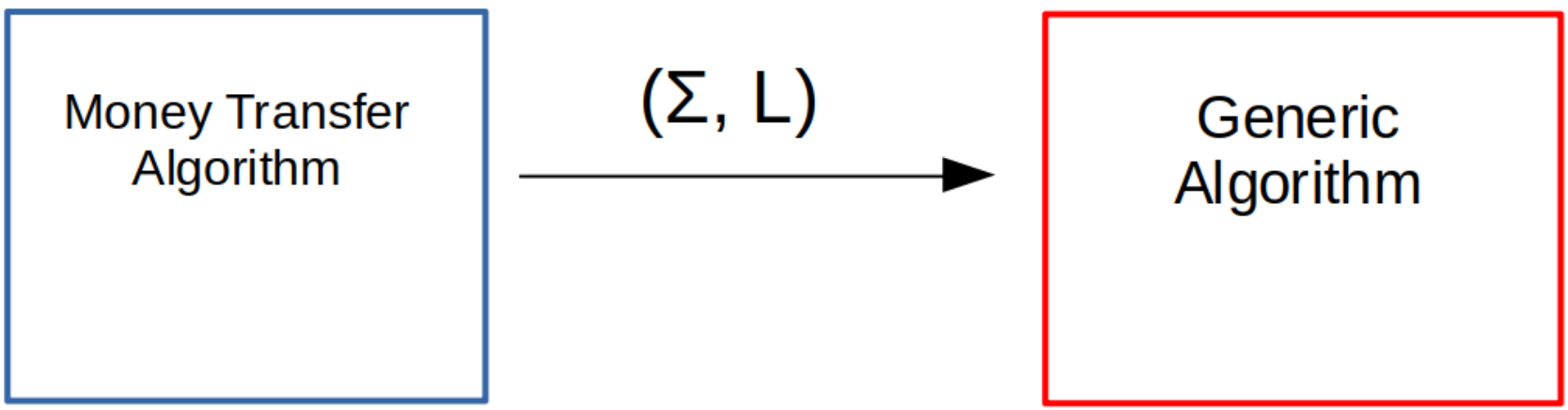
Lucie Guillou

Ecole normale supérieure de Rennes

INRIA Rennes : Team WIDE
François Taïani, Davide Frey, Michel Raynal

November 18, 2020

Introduction



Overview

State of the Art and Motivation

Why now ?

State of the Art and Motivation

Why now ?

- 2019 : Important result on money transfer complexity [1]

State of the Art and Motivation

Why now ?

- 2019 : Important result on money transfer complexity [1]
- Money Transfer made simple [2]

State of the Art and Motivation

Why now ?

- 2019 : Important result on money transfer complexity [1]
- Money Transfer made simple [2]

Why this subject ?

State of the Art and Motivation

Why now ?

- 2019 : Important result on money transfer complexity [1]
- Money Transfer made simple [2]

Why this subject ?

- Simple algorithm

State of the Art and Motivation

Why now ?

- 2019 : Important result on money transfer complexity [1]
- Money Transfer made simple [2]

Why this subject ?

- Simple algorithm
- Characterize a larger class of complexity

Definitions and model

- n processes

Definitions and model

- n processes
- *An asynchronous distributed system*

Definitions and model

- n processes
- *An asynchronous distributed system*
- *A crash tolerant algorithm*

Definitions and model

- n processes
- *An asynchronous distributed system*
- *A crash tolerant algorithm*
- *A Byzantine tolerant algorithm*

About the algorithm

A money transfer algorithm

About the algorithm

A money transfer algorithm

- One account per process

About the algorithm

A money transfer algorithm

- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)

About the algorithm

A money transfer algorithm

- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)
- *Specific* transfer operations

About the algorithm

A money transfer algorithm

- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)
- *Specific* transfer operations

No double spending !

About the algorithm

A money transfer algorithm

- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)
- *Specific* transfer operations

No double spending !

A process :

About the algorithm

A money transfer algorithm

- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)
- *Specific* transfer operations

No double spending !

A process :

- checks validity of each transfer operation,

About the algorithm

A money transfer algorithm

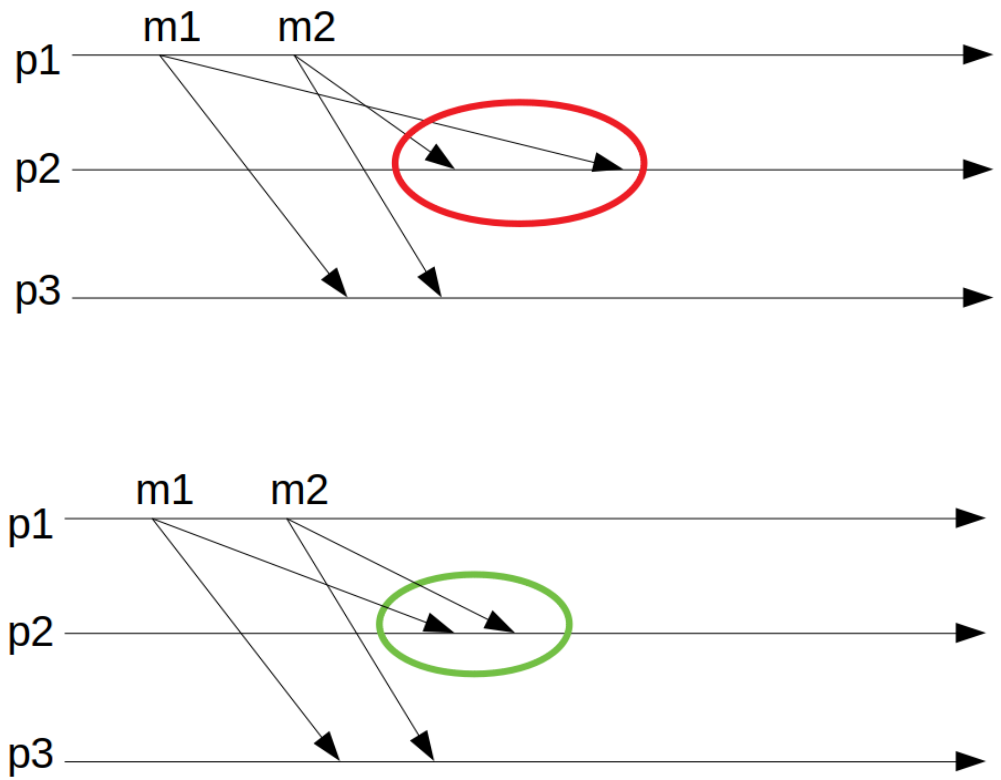
- One account per process
- Broadcast algorithm (Byzantine tolerant / crash tolerant)
- *Specific* transfer operations

No double spending !

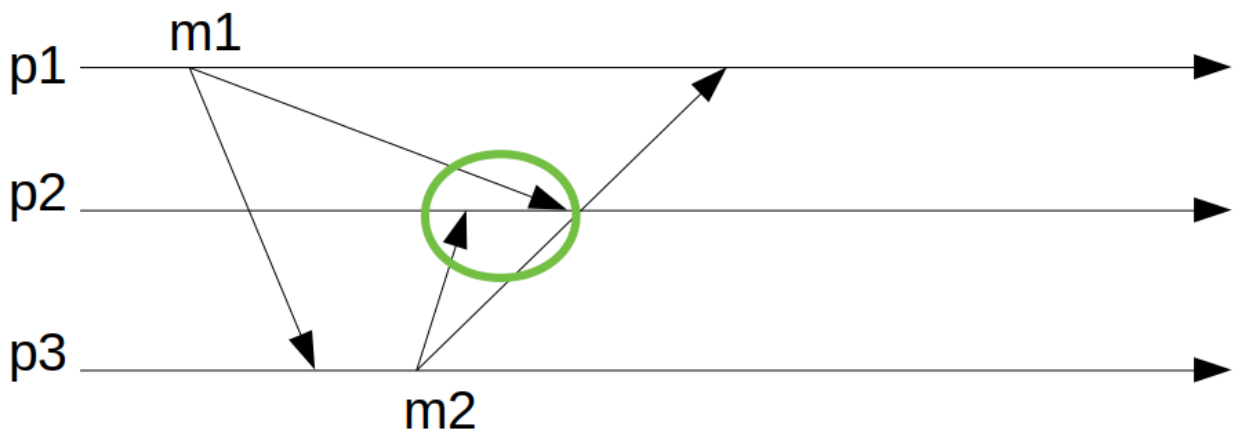
A process :

- checks validity of each transfer operation,
- orders all transfer operations of one account.

FIFO order between messages of a same process...



... But not between messages of different processes.



My contribution

Aim :

- Alphabet and language over operations
- Generic algorithm based on the Money Transfer algorithm
- Applications, language's instances

Alphabet definition

Alphabet over operations Σ

$n + 1$ sets of operations :

Alphabet definition

Alphabet over operations Σ

$n + 1$ sets of operations :

- C : common operation.

Alphabet definition

Alphabet over operations Σ

$n + 1$ sets of operations :

- C : common operation.
- $(O_i)_{1 \leq i \leq n}$: specifics operations.

Alphabet definition

Alphabet over operations Σ

$n+1$ sets of operations :

- C : common operation.
- $(O_i)_{1 \leq i \leq n}$: specifics operations.

$\Sigma := C \cup \bigcup_{1 \leq i \leq n} O_i$

Alphabet definition

Alphabet over operations Σ

$n+1$ sets of operations :

- C : common operation.
- $(O_i)_{1 \leq i \leq n}$: specifics operations.

$\Sigma := C \cup \bigcup_{1 \leq i \leq n} O_i$

Example : Money transfer : $C = \emptyset,\ O_i = \{\mathrm{trf}(p_i, n, p_j)\}_{j,n \in \mathbb{N}}$.

Commutation relation

Independence relations

- Commutation relation $I = \Sigma \setminus (O_i \times O_i)_i$

Commutation relation

Independence relations

- Commutation relation $I = \Sigma \setminus (O_i \times O_i)_i$
- Equivalence relation $\sim_I$

Commutation relation

Independence relations

- Commutation relation $I = \Sigma \setminus (O_i \times O_i)_i$
- Equivalence relation $\sim_I$

Example : Money transfer

Commutation relation

Independence relations

- Commutation relation $I = \Sigma \setminus (O_i \times O_i)_i$
- Equivalence relation $\sim_I$

Example : Money transfer

- $(\mathrm{trf}(p_1, 10, p_2), \mathrm{trf}(p_2, 10, p_1)) \in I$

Commutation relation

Independence relations

- Commutation relation $I = \Sigma \setminus (O_i \times O_i)_i$
- Equivalence relation $\sim_I$

Example : Money transfer

- $(\mathrm{trf}(p_1, 10, p_2), \mathrm{trf}(p_2, 10, p_1)) \in I$
- $(\mathrm{trf}(p_1, 10, p_2), \mathrm{trf}(p_1, 10, p_3)) \notin I$

Language of legals executions

Language properties

L's properties :

Language of legals executions

Language properties

L's properties :

1. Legal past $\Rightarrow$ prefix closed.

Language of legals executions

Language properties

L's properties :

1. Legal past $\Rightarrow$ prefix closed.
2. Common operations always legal $\Rightarrow$ $LC^* \subseteq L$.

Language of legals executions

Language properties

L's properties :

1. Legal past $\Rightarrow$ prefix closed.
2. Common operations always legal $\Rightarrow$ $LC^* \subseteq L$.
3. Same legal future for two equivalents and legals executions.

Language of legals executions

Language properties

L's properties :

1. Legal past $\Rightarrow$ prefix closed.
2. Common operations always legal $\Rightarrow LC^* \subseteq L$.
3. Same legal future for two equivalents and legals executions.
4. $o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

$\text{account}_1 = 10$, $\text{account}_2 = 10$, $\text{account}_3 = 10$.

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

$\text{account}_1 = 10$, $\text{account}_2 = 10$, $\text{account}_3 = 10$.

$v = \text{trf}(p_1, 5, p_2) \cdot \text{trf}(p_3, 5, p_1)$, $o = \text{trf}(p_1, 10, p_2)$,

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

$\text{account}_1 = 10$, $\text{account}_2 = 10$, $\text{account}_3 = 10$.
$v = \text{trf}(p_1, 5, p_2) \cdot \text{trf}(p_3, 5, p_1)$, $o = \text{trf}(p_1, 10, p_2)$,
$w = \text{trf}(p_2, 5, p_3) \cdot \text{trf}(p_3, 5, p_1) \cdot \text{trf}(p_1, 5, p_2)$.

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

$\text{account}_1 = 10$, $\text{account}_2 = 10$, $\text{account}_3 = 10$.
$v = \text{trf}(p_1, 5, p_2) \cdot \text{trf}(p_3, 5, p_1)$, $o = \text{trf}(p_1, 10, p_2)$,
$w = \text{trf}(p_2, 5, p_3) \cdot \text{trf}(p_3, 5, p_1) \cdot \text{trf}(p_1, 5, p_2)$.
$w \cdot o \in L$

Illustrations of the 4th property

Property

$o \in O_i$, $v \in L$, s.t $v \cdot o \in L$. For $w \in L$, if w contains v (not necessary in the same order) and $(w \setminus v) \cap O_i = \emptyset$, then $w \cdot o \in L$.

Example on Money Transfer

$\text{account}_1 = 10$, $\text{account}_2 = 10$, $\text{account}_3 = 10$.
$v = \text{trf}(p_1, 5, p_2) \cdot \text{trf}(p_3, 5, p_1)$, $o = \text{trf}(p_1, 10, p_2)$,
$w = \text{trf}(p_2, 5, p_3) \cdot \text{trf}(p_3, 5, p_1) \cdot \text{trf}(p_1, 5, p_2)$.
$w \cdot o \in L$

Additional operations in w do not reduce p_1's balance.

The generic algorithm

Similarities :

The generic algorithm

Similarities :

- Broadcast algorithm

The generic algorithm

Similarities :

- Broadcast algorithm
- Order between operations of one *specific* set.

The generic algorithm

Similarities :

- Broadcast algorithm
- Order between operations of one *specific* set.

Differencies :

The generic algorithm

Similarities :

- Broadcast algorithm
- Order between operations of one *specific* set.

Differencies :

- Legacy oracle

The generic algorithm

Similarities :

- Broadcast algorithm
- Order between operations of one *specific* set.

Differencies :

- Legacy oracle
- Keep the current sequence execution received

The generic algorithm

Similarities :

- Broadcast algorithm
- Order between operations of one *specific* set.

Differencies :

- Legacy oracle
- Keep the current sequence execution received

Difference between each executions perceived but same legacy.

Applications

- Multi-set distributed

Applications

- Multi-set distributed
- Petri net but with some restrictions

Applications

- Multi-set distributed
- Petri net but with some restrictions
- Work stealing... Very restricted

Conclusion

The following...

- a publication ?
- a better modelization of the object work stealing ?

[1] R. Guerraoui, P. Kuznetsov. The Consensus Number of a Cryptocurrency. *PODC*, 2019.

[2] A. Auvolat, D. Frey, M. Raynal, F.Taïani. Money Transfer Made Simple. 2020.