r-semi-groups: a generic approach for designing stabilizing silent tasks

Bertrand Ducourthial

Laboratoire Heudiasyc (UMR UTC-CNRS 6599)
Computer Science Dpt
Université de Technologie de Compiègne

SSS'2007
Ninth International Symposium on Stabilization, Safety, and Security of Distributed Systems,
14th - 16th November 2007, Paris, France

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Université de Technologie Compiègne

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

r-semi-groups: a generic approach for designing stabilizing silent tasks

1 Contribution

2 Generic approach with r-semi-groups

3 Main results

4 Using r-semi-groups

5 Conclusion

Summary

1 Contribution

2 Generic approach with r-semi-groups

3 Main results

4 Using r-semi-groups

5 Conclusion

Contribution

- *Static* versus *dynamic* task
  - a static task is specified by a result
    - specification $\equiv$ predicats on configurations
    - silent algorithms
      (without self-stabilization consideration)
    - example: leader election
  - a dynamic task is specified by a behavior
    - specification $\equiv$ predicats on executions
    - example: token circulation

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

Contribution

- *Static* versus *dynamic* task
- Our contribution: r-semi-groups
  - complete framework to design
    - static tasks
    - self-stabilizing static tasks

Contribution

- *Static* versus *dynamic* task
- Our contribution: r-semi-groups
  - complete framework to design
    - static tasks
    - self-stabilizing static tasks
- Interest
  - design of algorithms
  - local conditions for ensuring global (self-)stabilization
  - easy proof by reusing generic results
    proofs established for all r-semi-groups

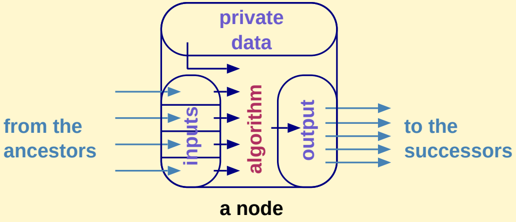
Related work

- Generic approaches to prove self-stabilizing properties of distributed algorithms
  [Arora93, Afek98, Theel00, Gouda03, Oehlerking05]
  not as generic as r-semi-groups
- Path algebra, max-plus algebra
  [Aho74,Gondran79,Baccelli92]
  two laws, relation with algorithms more complex
- Relations between algebraic structures and computations on networks
  [Bilardi90]
  not a framework to design distributed application

Summary

1. Contribution
2. Generic approach with r-semi-groups
   - Algorithm $\equiv$ operator
   - A quick reminder on algebra
   - What kind of operators?
3. Main results
4. Using r-semi-groups
5. Conclusion

Thinking the algorithm as an operator

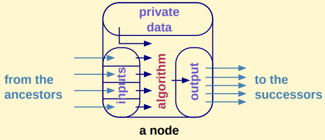


Static task:

algorithm $\equiv$

$$\text{output} \leftarrow f(\text{private data}, \text{input}[1], \ldots, \text{input}[n])$$

Thinking the algorithm as an operator



Algorithm (message passing):

$\mathcal{R}_1$ Upon receipt of a message m sent by u:

if $m \neq \text{input}[u]$, then

input$[u] \leftarrow m$

output $\leftarrow f(\text{private data}, \text{input}[1], \ldots, \text{input}[n])$

/* local computation with the local r-operator */

send(output) to the neighbors

end if

$\mathcal{R}_2$ Upon timeout expiration:

output $\leftarrow f(\text{private data}, \text{input}[1], \ldots, \text{input}[n])$

send(output) to the neighbors

reset the timeout

A quick reminder on algebra

magma $(\mathbb{S}, \diamond)$

A quick reminder on algebra

$$\begin{tikzcd}
\textbf{magma } (\mathbb{S}, \diamond) \arrow[d, "\diamond \text{ associative}"] \\
\textbf{monoid}
\end{tikzcd}$$

A quick reminder on algebra

$$
\begin{array}{ll}
\textbf{magma } (\mathbb{S}, \diamond) & \\
\downarrow \ \diamond \text{ associative} & \\
\textbf{monoid} & \\
\downarrow \ e_\diamond \text{ identity element (on the left and the right)} & \\
\textbf{semi-group} & \preceq \text{ preorder relation} \\
 & a \preceq b \equiv \exists c, a = b \diamond c
\end{array}
$$

A quick reminder on algebra

magma $(\mathbb{S}, \diamond)$

↓ $\diamond$ associative

monoid

↓ $e_\diamond$ identity element (on the left and the right)

semi-group — $\preceq$ preorder relation $a \preceq b \equiv \exists c, a = b \diamond c$

↙ every element is cancellative $x \diamond y = x \diamond z \Rightarrow y = z$

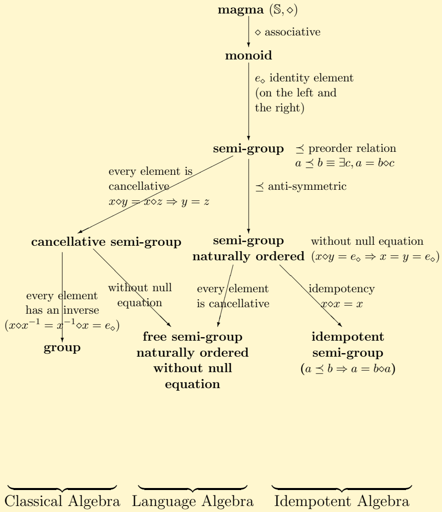
cancellative semi-group

↓ $\preceq$ anti-symmetric

semi-group naturally ordered — without null equation $(x \diamond y = e_\diamond \Rightarrow x = y = e_\diamond)$

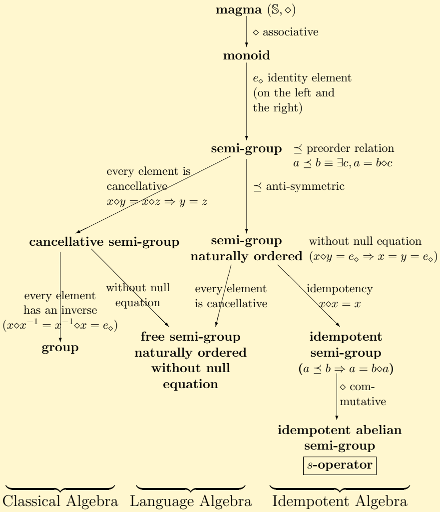
Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

A quick reminder on algebra



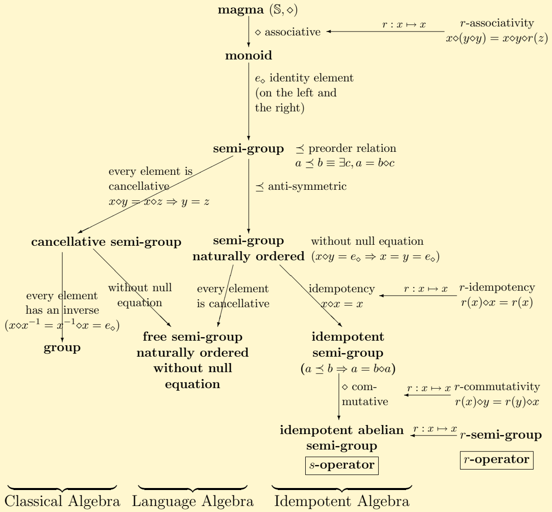
A quick reminder on algebra

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion



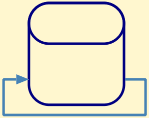
Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

A quick reminder on algebra



What kind of operators?

- A sort of idempotency is needed



Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

What kind of operators?

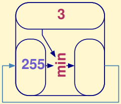
- A sort of idempotency is needed
- Idempotent Abelian semi-groups s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent

What kind of operators?

- A sort of idempotency is needed
- Idempotent Abelian semi-groups — s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent
- What if transient failure?
  - could stabilize on an illegitimate value [DistComp01]
  - self-stabilizing on acyclic networks

What kind of operators?

- A sort of idempotency is needed
- Idempotent Abelian semi-groups s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent
- What if transient failure?
  - could stabilize on an illegitimate value [DistComp01]
  - self-stabilizing on acyclic networks

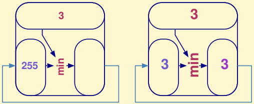


initial configuration

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

What kind of operators?

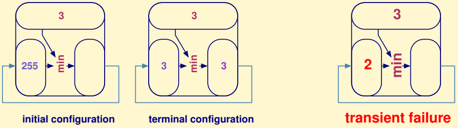
- ○ A sort of idempotency is needed
- Idempotent Abelian semi-groups — s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent
- What if transient failure?
  - could stabilize on an illegitimate value [DistComp01]
  - self-stabilizing on acyclic networks



initial configuration terminal configuration

What kind of operators?

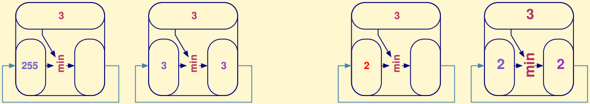
- A sort of idempotency is needed
- Idempotent Abelian semi-groups — s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent
- What if transient failure?
  - could stabilize on an illegitimate value [DistComp01]
  - self-stabilizing on acyclic networks



Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

What kind of operators?

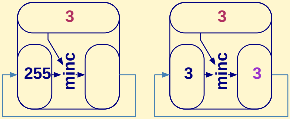
- A sort of idempotency is needed
- Idempotent Abelian semi-groups s-operator
  - $\wedge$, $\vee$, min, max, gcd, lcm, $\cap$, $\cup$... [Tel91]
  - associative, commutative and idempotent
- What if transient failure?
  - could stabilize on an illegitimate value [DistComp01]
  - self-stabilizing on acyclic networks



initial configuration | terminal configuration | transient failure | illegitimate terminal config.

What kind of operators?

- Some operators tolerate transient failures
  - $\text{minc}(x, y) = \min(x, y + 1)$
    defined on $\mathbb{N} \cup \{+\infty\}$ or $\{0, \ldots, 255\}$...

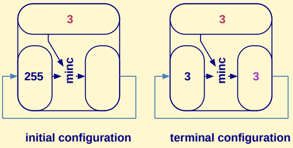


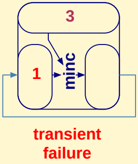
initial configuration **terminal configuration**

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

What kind of operators?

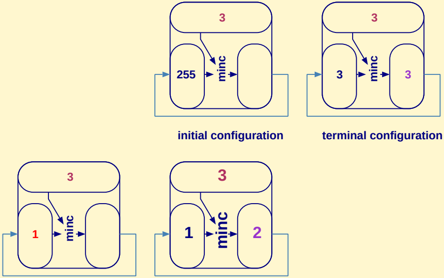
- Some operators tolerate transient failures
  - $\mathrm{minc}(x, y) = \min(x, y + 1)$
    defined on $\mathbb{N} \cup \{+\infty\}$ or $\{0, \ldots, 255\}$...





What kind of operators?

- Some operators tolerate transient failures
  - $\mathrm{minc}(x, y) = \min(x, y + 1)$
    defined on $\mathbb{N} \cup \{+\infty\}$ or $\{0, \ldots, 255\}$...



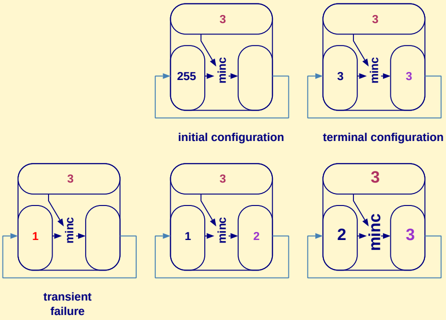
transient failure

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

utc
Université de Technologie Compiègne

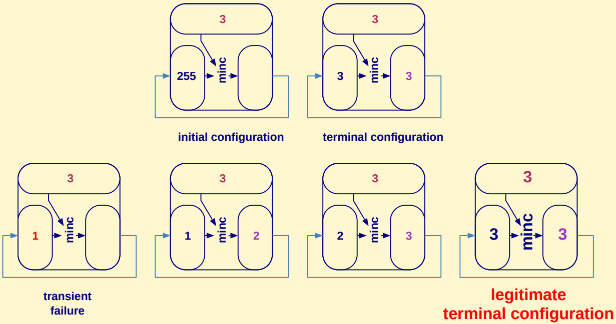
What kind of operators?

- Some operators tolerate transient failures
  - $\mathrm{minc}(x, y) = \min(x, y + 1)$
    defined on $\mathbb{N} \cup \{+\infty\}$ or $\{0, \ldots, 255\}$...



What kind of operators?

- Some operators tolerate transient failures
  - $\mathrm{minc}(x, y) = \min(x, y + 1)$
    defined on $\mathbb{N} \cup \{+\infty\}$ or $\{0, \ldots, 255\}$...



What kind of operators

Summary of properties

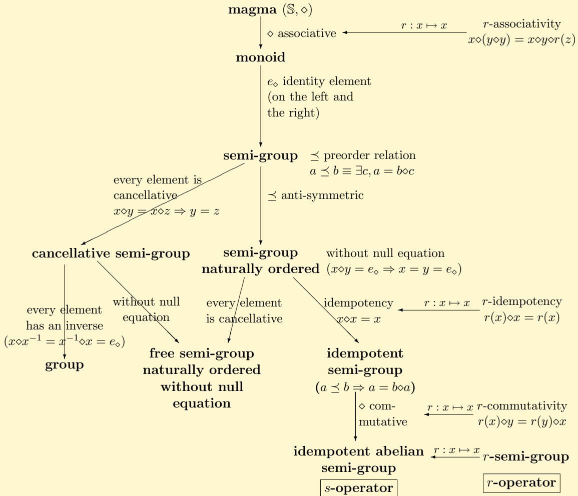
Operator	Properties of the algorithm	Proof
associative and commutative	silent task if there is no circuit	
associative, commutative and idempotent	silent task not self-stabilizing	Tel91 DistComp01
idempotent r-operator	silent task	SIROCCO98, DistComp01
strictly idemp. r-operator with total order	self-stabilizing - read-write demon, shared memory (registers) - unreliable messages passing	DistComp01 SSS05,JACIC06
strictly idempotent r-operator with partial order	self-stabilizing, shared memory, fully distributed demon	TCS03

Summary

1. Contribution
2. Generic approach with r-semi-groups
3. Main results
4. Using r-semi-groups
5. Conclusion

r-semi-group

A generalization of the idempotent Abelian semi-group with interesting properties for computation in networks



Construction of the r-operators
Required properties

- Weak left cancellation
  - idempotency vs. cancellation

Construction of the r-operators
Required properties

- Weak left cancellation
  - idempotency vs. cancellation
  - $\forall x, y, z \in \mathbb{S}, \quad (y = z) \Rightarrow (x \diamond y = x \diamond z)$
    always true

Construction of the r-operators
Required properties

- Weak left cancellation
  - idempotency vs. cancellation
  - $\forall x, y, z \in \mathbb{S}, \quad (y = z) \Rightarrow (x \diamond y = x \diamond z)$
    always true
  - $\forall x, y, z \in \mathbb{S}, \quad (x \diamond y = x \diamond z) \Rightarrow (y = z)$
    cancellation
    false for min on $\mathbb{N}$ (consider $x = 2,\ y = 3,\ z = 4$)

Construction of the r-operators
Required properties

- Weak left cancellation
  - idempotency vs. cancellation
  - $\forall x, y, z \in \mathbb{S}, \quad (y = z) \Rightarrow (x \diamond y = x \diamond z)$
    always true
  - $\forall x, y, z \in \mathbb{S}, \quad (x \diamond y = x \diamond z) \Rightarrow (y = z)$
    cancellation
    false for min on $\mathbb{N}$ (consider $x = 2$, $y = 3$, $z = 4$)
  - $\forall y, z \in \mathbb{S}, \quad (\forall x \in \mathbb{S}, x \diamond y = x \diamond z) \Rightarrow y = z$
    true for min on $\mathbb{N}$

Construction of the r-operators
Required properties

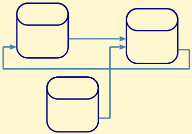
- Weak left cancellation
  - idempotency vs. cancellation
  - $\forall x, y, z \in \mathbb{S}, \quad (y = z) \Rightarrow (x \diamond y = x \diamond z)$
    always true
  - $\forall x, y, z \in \mathbb{S}, \quad (x \diamond y = x \diamond z) \Rightarrow (y = z)$
    cancellation
    false for min on $\mathbb{N}$ (consider $x = 2,\ y = 3,\ z = 4$)
  - $\forall y, z \in \mathbb{S}, \quad (\forall x \in \mathbb{S}, x \diamond y = x \diamond z) \Rightarrow y = z$
    true for min on $\mathbb{N}$

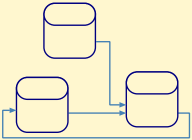
$\leadsto$ weak left cancellation
min is weak left cancellative

Construction of the r-operators

Required properties

- ○ Weak left cancellation
- Local topology awareness





$$\text{private data} \triangleleft \text{input}[1] \triangleleft \text{input}[2] =$$
$$\text{private data} \triangleleft \text{input}[2] \triangleleft \text{input}[1]$$

$\rightsquigarrow$ rank 2 commutativity

min is rank 1 commutative (commutative)

Construction of the r-operators

Required properties

- ○ Weak left cancellation
- ○ Local topology awareness
- • Termination

removing doubles in expressions:

private data $\triangleleft$ input $\triangleleft$ input $=$
private data $\triangleleft$ input

$\rightsquigarrow$ rank 2 idempotency

min is rank 1 idempotent (idempotent)

Construction of the r-operators

Generalization of the semi-group

- *s*-operator $\oplus$ (infimum) *e.g.*, min

  idempotent Abelian semi-group

associative	$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
commutative	$x \oplus y = y \oplus x$
idempotent	$x \oplus x = x$
identity element	$x \oplus e_{\oplus} = x$

Construction of the r-operators

Generalization of the semi-group

- s-operator $\oplus$ (infimum) *e.g.*, min
- r-operator *e.g.*, $\mathrm{minc}(x, y) = \min(x, y + 1)$

r-semi-group

$\lhd$ is an r-operator on $\mathbb{S}$ if there exists an endomorphism $r : \mathbb{S} \to \mathbb{S}$ such that $\lhd$ is :

$$\begin{array}{lrcccl}
r\text{-associative} & (x \lhd y) & \lhd & r(z) & = & x \lhd (y \lhd z) \\
r\text{-commutative} & r(x) & \lhd & y & = & r(y) \lhd x \\
r\text{-idempotent} & r(x) & \lhd & x & = & r(x) \\
\text{right identity elt} & x & \lhd & e_{\lhd} & = & x
\end{array}$$

$(\mathbb{S}, \lhd)$ admits an order relation

Construction of the r-operators
Generalization of the semi-group

- s-operator $\oplus$ (infimum) *e.g.*, min
- r-operator *e.g.*, $\mathrm{minc}(x, y) = \min(x, y + 1)$
- idempotency
  - idempotent r-operator:
    $\forall x \in \mathbb{S}, \quad x \triangleleft x = x$
    $\rightsquigarrow x \preceq_{\triangleleft} r(x)$
    usefull for termination
  - strictly idempotent r-operator:
    $x \prec_{\triangleleft} r(x)$
    usefull for self-stabilization

r-semi-group: a summary

magma $(\mathbb{S}, \triangleleft)$

$r : \mathbb{S} \to \mathbb{S}$

$\triangleleft$ weak left cancellative

$e_\triangleleft$ right identity element

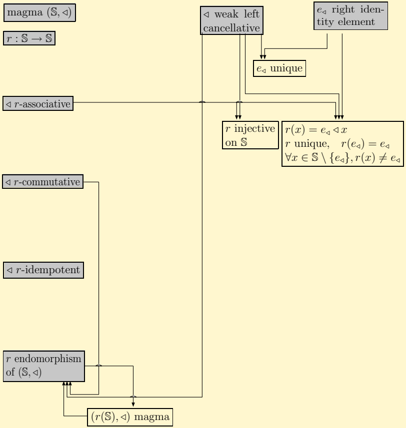
$\triangleleft$ r-associative

$\triangleleft$ r-commutative

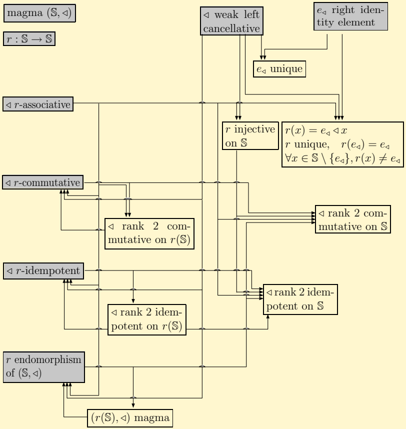
$\triangleleft$ r-idempotent

r endomorphism of $(\mathbb{S}, \triangleleft)$

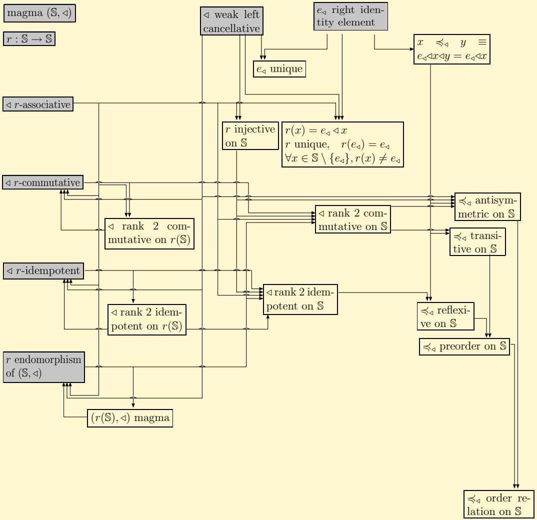
r-semi-group: a summary



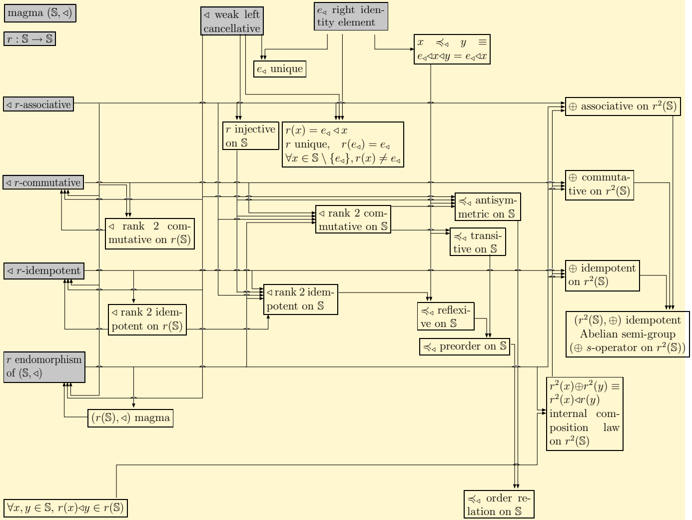
r-semi-group: a summary



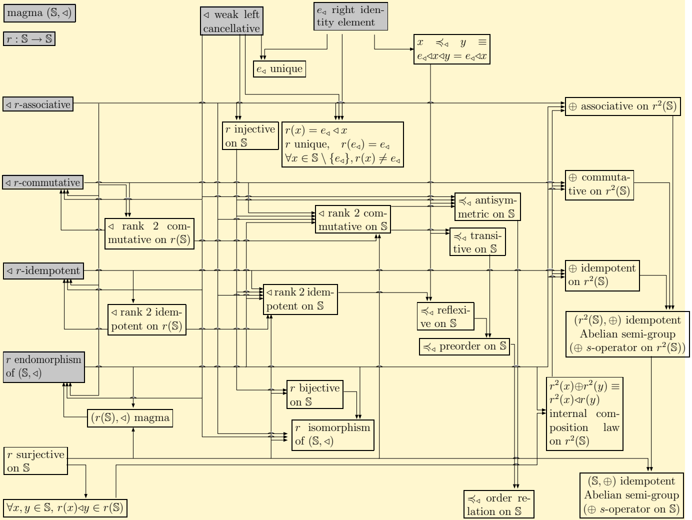
r-semi-group: a summary



r-semi-group: a summary



r-semi-group: a summary



Summary

1 Contribution

2 Generic approach with r-semi-groups

3 Main results

4 Using r-semi-groups
   Method
   HOWTO r-semi-group

5 Conclusion

Thinking the algorithm as an operator

Method:

- Data exchanged, data owned by a node, set $\mathbb{S}$
- Given two data, choice of the best
  - $\rightsquigarrow$ order relation $\preceq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{S}, \oplus)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Data exchanged, data owned by a node, set $\mathbb{S}$
- Given two data, choice of the best
  - $\rightsquigarrow$ order relation $\preceq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{S}, \oplus)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Data exchanged, data owned by a node, set $\mathbb{S}$
  distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the best
  $\rightsquigarrow$ order relation $\preceq$
  $\rightsquigarrow$ idempotent semi-group $(\mathbb{S}, \oplus)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  $\rightsquigarrow$ idempotency
  $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  $\rightsquigarrow$ strict idempotency
  $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the best
  - $\rightsquigarrow$ order relation $\preceq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{S}, \oplus)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the best — smallest
  - $\rightsquigarrow$ order relation $\preceq$ — $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{S}, \oplus)$ — $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\rightsquigarrow$ order relation $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto r(x)$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\rightsquigarrow$ order relation $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto r(x)$
  at the arrival, one hop more: $x \mapsto x + 1$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\rightsquigarrow$ order relation $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto x + 1$
- r endomorphism of $(\mathbb{S}, \oplus)$
  - $\rightsquigarrow$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  $\rightsquigarrow$ order relation $\leq$
  $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto x + 1$
- r endomorphism of $(\mathbb{S}, \oplus)$
  $\rightsquigarrow$ r-semi-group
  $x \mapsto x + 1$ endomorphism of $(\mathbb{N} \cup \{\infty\}, \mathrm{minc})$
- $x \preceq r(x)$
  $\rightsquigarrow$ idempotency
  $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  $\rightsquigarrow$ strict idempotency
  $\rightsquigarrow$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\leadsto$ order relation $\leq$
  - $\leadsto$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto x + 1$
- $\mathrm{minc}(x, y) = \min(x, y + 1)$ r-operator
  $(\mathbb{N} \cup \{\infty\}, \mathrm{minc})$ r-semi-group
- $x \preceq r(x)$
  - $\leadsto$ idempotency
  - $\leadsto$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\leadsto$ strict idempotency
  - $\leadsto$ self-stabilizing algorithm

Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\rightsquigarrow$ order relation $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto x + 1$
- $\text{minc}(x, y) = \min(x, y + 1)$ r-operator
  $(\mathbb{N} \cup \{\infty\}, \text{minc})$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x \preceq r(x)$ and $r(x) \neq x$
  - $\rightsquigarrow$ strict idempotency
  - $\rightsquigarrow$ self-stabilizing algorithm

  $x < r(x)$

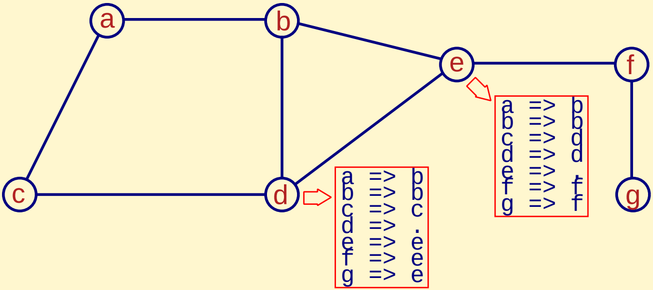
Thinking the algorithm as an operator

Method: example: distance from a node u

- Distance from u, u owns 0, others ∞, $\mathbb{N} \cup \{\infty\}$
- Given two data, choice of the smallest
  - $\rightsquigarrow$ order relation $\leq$
  - $\rightsquigarrow$ idempotent semi-group $(\mathbb{N}, \min)$
- Transformation of the incoming data: $x \mapsto x + 1$
- $\text{minc}(x, y) = \min(x, y + 1)$ r-operator $(\mathbb{N} \cup \{\infty\}, \text{minc})$ r-semi-group
- $x \preceq r(x)$
  - $\rightsquigarrow$ idempotency
  - $\rightsquigarrow$ stabilization (termination)
- $x < r(x)$
  - $\rightsquigarrow$ minc strictly idempotent
  - $\rightsquigarrow$ self-stabilizing distance computation alg.
- Q.E.D.

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

HOWTO r-semi-group

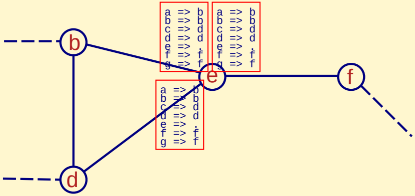


- Example: how to design a self-stabilizing algorithm for routing tables construction?

HOWTO r-semi-group

Routing tables construction 1/2

- A node periodically sends its local informations to its neighbors

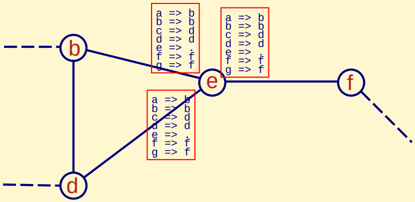


Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

HOWTO r-semi-group

Routing tables construction 1/2

- A node periodically sends its local informations to its neighbors

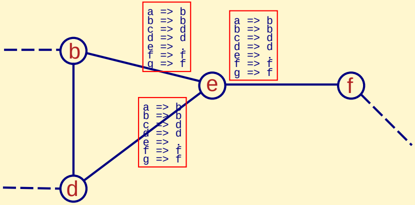


Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

HOWTO r-semi-group

Routing tables construction 1/2

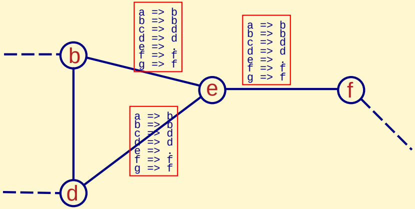
- A node periodically sends its local informations to its neighbors



HOWTO r-semi-group

Routing tables construction 1/2

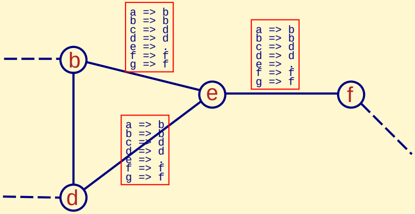
- A node periodically sends its local informations to its neighbors



HOWTO r-semi-group

Routing tables construction 1/2

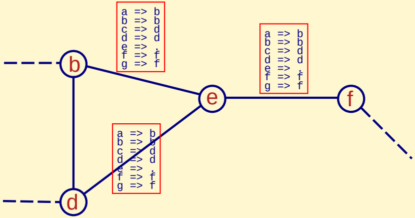
- A node periodically sends its local informations to its neighbors



HOWTO r-semi-group

Routing tables construction 1/2

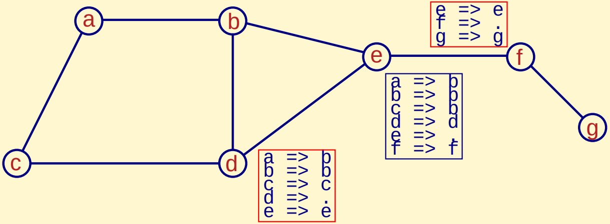
- A node periodically sends its local informations to its neighbors



HOWTO r-semi-group

Routing tables construction 2/2

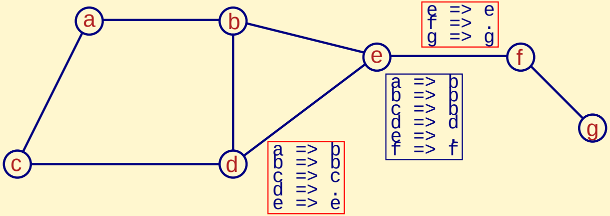
- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node ⇝ add
  - if it has a worse information ⇝ update



HOWTO r-semi-group

Routing tables construction 2/2

- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node $\rightsquigarrow$ add
  - if it has a worse information $\rightsquigarrow$ update

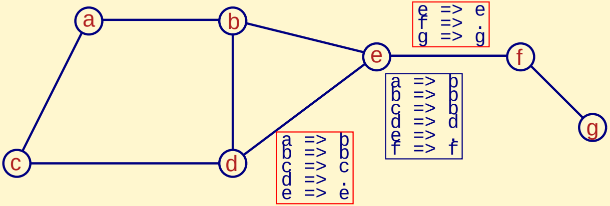


Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

HOWTO r-semi-group

Routing tables construction 2/2

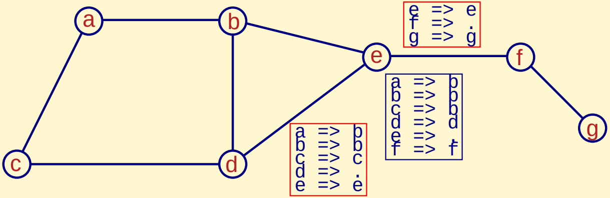
- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node $\rightsquigarrow$ add
  - if it has a worse information $\rightsquigarrow$ update



HOWTO r-semi-group

Routing tables construction 2/2

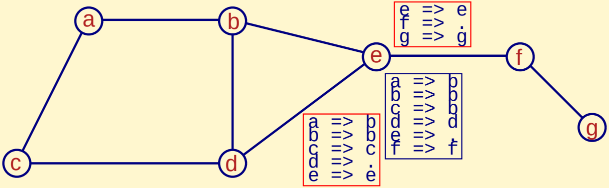
- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node $\rightsquigarrow$ add
  - if it has a worse information $\rightsquigarrow$ update



HOWTO r-semi-group

Routing tables construction 2/2

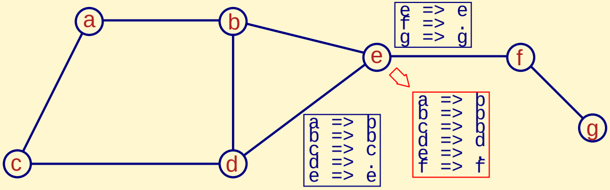
- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node ⇝ add
  - if it has a worse information ⇝ update



CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

HOWTO r-semi-group
Routing tables construction 2/2

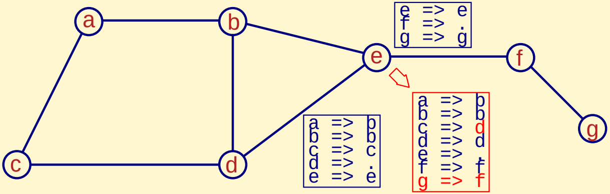
- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node $\rightsquigarrow$ add
  - if it has a worse information $\rightsquigarrow$ update



HOWTO r-semi-group

Routing tables construction 2/2

- When a node receives an information regarding a destination node, it checks:
  - if it has no information for this node $\rightsquigarrow$ add
  - if it has a worse information $\rightsquigarrow$ update



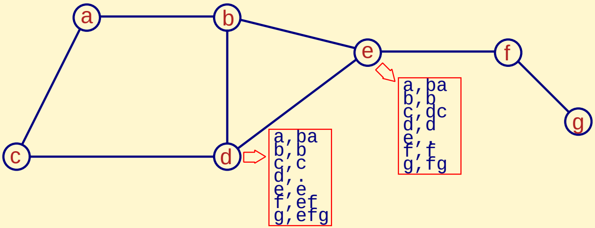
HOWTO r-semi-group
1. Data format

- Information regarding a destination:
  - next hop to reach the destination
  - information regarding the path

    to select the best next hop
- Entry in the routing table:

  [destination, path to the destination]
- Local information: list of entries

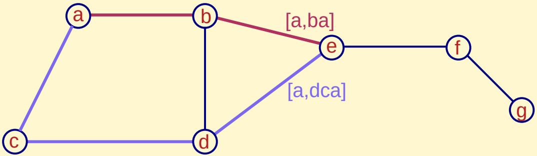
  example: $[a, ba], [b, b], [c, c], [d, \emptyset], [e, e], [f, ef], [g, efg]$



HOWTO r-semi-group
2. Order relation on the data

- Choosing the best path to a given destination
  - operator *best* on the paths to a common node
    return (for instance) the shortest path $\oplus$
  - example (shortest path): $ba \oplus dca = ba$

  - operator *best* on the similar tables entries
    return the entry with the best path $\boxplus$
  - example (shortest path):
    $[a, ba] \boxplus [a, dca] = [a, ba \oplus dca] = [a, ba]$



HOWTO r-semi-group

2. Order relation on the data

- Choosing the best path to a given destination
- Building a new table from two tables
  - no more than one path for a destination
    $\rightsquigarrow$ choosing the best
  - operator fusion on the list of entries: $\uplus$
    - union of the list
    - choice of the best entry in case of same destination — using the operator *best*
  - example:

$$
\begin{aligned}
& ([a,a],[b,b],[c,abc]) \uplus ([a,da],[c,dc],[d,d]) \\
= & ([a,a] \boxplus [a,da],[b,b],[c,abc] \boxplus [c,dc],[d,d]) \\
= & ([a,a \oplus da],[b,b],[c,abc \oplus dc],[d,d])
\end{aligned}
$$

HOWTO r-semi-group

2. Order relation on the data

- Choosing the best path to a given destination
- Building a new table from two tables
- Properties of the operators
  - operator best on the paths $\oplus$
    - associative
      $ba \oplus (dca \oplus bcda) = (ba \oplus dca) \oplus bcda$
    - commutative $ba \oplus dca = dca \oplus ba$
    - idempotent $ba \oplus ba = ba$
  - operator best on the entries $\boxplus$ $\rightsquigarrow$ idem
    $[a, ba] \boxplus [a, dca] = [a, ba \oplus dca] = [a, ba]$
  - operator fusion on the lists of entries $\uplus$ $\rightsquigarrow$ idem
    $([a, a], [b, b], [c, abc]) \uplus ([a, da], [c, dc], [d, d])$
    $= ([a, a] \boxplus [a, da], [b, b], [c, abc] \boxplus [c, dc], [d, d])$
    $= ([a, a \oplus da], [b, b], [c, abc \oplus dc], [d, d])$

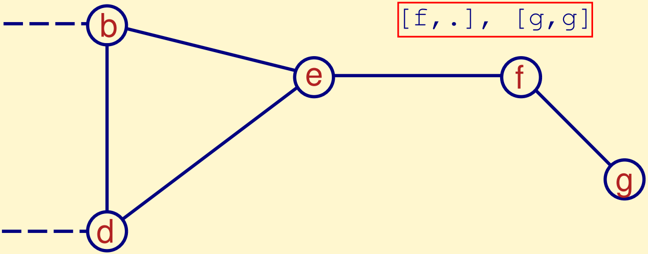
$\rightsquigarrow$ $\uplus$ s-operator

idempotent Abelian semi-group

HOWTO r-semi-group

3. Transformation of the incoming data

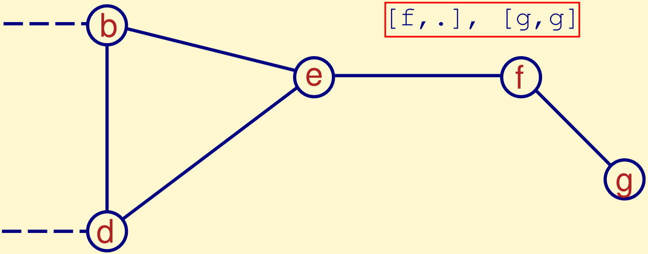
- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

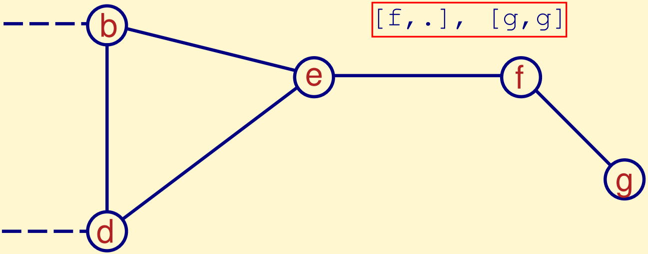
- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

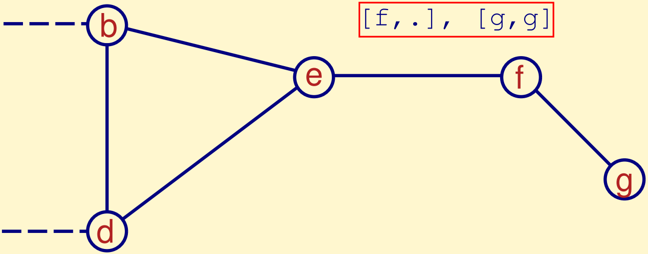
- Adding the sender at the begining of each path:
  if the node f sends the list $([f, .], [g, g])$, this list becomes $([f, f], [g, fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

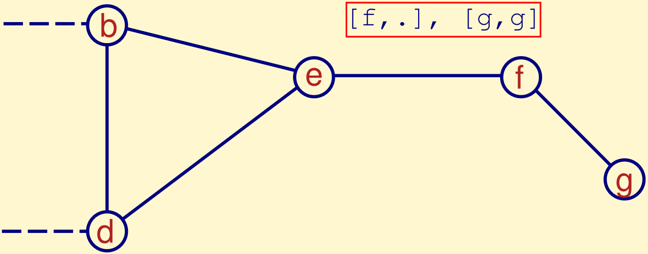
- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

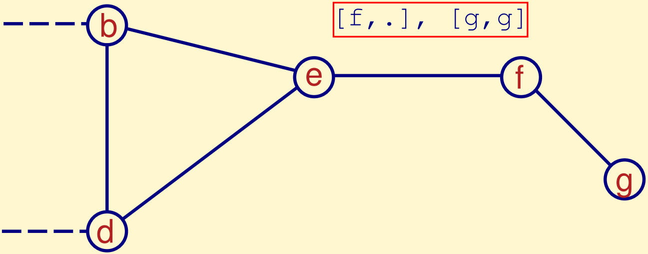
- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival

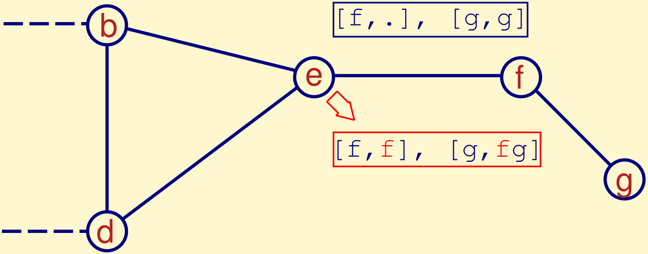


Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

HOWTO r-semi-group

3. Transformation of the incoming data

- Adding the sender at the begining of each path:
  if the node f sends the list $([f,.],[g,g])$, this list becomes $([f,f],[g,fg])$ at the arrival



HOWTO r-semi-group

3. Transformation of the incoming data

- Adding the sender at the begining of each path:
- Transformation of the lists

  one application per link

  On neighbors of f:

$$\begin{array}{rcl} r : \{\text{lists}\} & \longrightarrow & \{\text{lists}\} \\ ([f,.],[g,g]) & \longmapsto & ([f,f],[g,fg]) \end{array}$$

HOWTO r-semi-group

3. Transformation of the incoming data

- Adding the sender at the begining of each path:
- Transformation of the lists
- r homomorphism

$$r(\text{list} \uplus \text{list}') = r(\text{list}) \uplus r(\text{list})$$

- $r(\text{list}) \uplus \text{list} = \text{list}$

$$\begin{aligned}
& r\,(([f,.],[g,g])) \uplus ([f,.],[g,g]) \\
= & ([f,f],[g,fg]) \uplus ([f,.],[g,g]) \\
= & ([f,f] \boxplus [f,.],[g,fg] \boxplus [g,g]) \\
= & ([f,f \oplus .],[g,fg \oplus g]) \\
= & ([f,.],[g,g])
\end{aligned}$$

HOWTO r-semi-group

4. Properties of the r-operator

- $\triangleleft$ r-operators on the lists:

$$\text{list} \triangleleft \text{list}' = \text{list} \quad \uplus \quad r(\text{list}')$$

- $\uplus$ s-operator
  $\rightsquigarrow$ $\preceq_{\uplus}$ order relation
  total/partial: depends on the operator best $\oplus$ on the paths
- $r(\text{list}) \uplus \text{list} = \text{list}$ and $r(\text{list}) \neq \text{list}$
  $\rightsquigarrow$ $\text{list} \prec_{\uplus} r(\text{list})$
  $\rightsquigarrow$ $\triangleleft$ strictly idempotent r-operator

HOWTO r-semi-group

4. Properties of the r-operator

- $\lhd$ r-operators on the lists:

$$\text{list} \lhd \text{list}' = \text{list} \quad \uplus \quad r(\text{list}')$$

- $\uplus$ s-operator
  $\leadsto$ $\preceq_{\uplus}$ order relation
  total/partial: depends on the operator best $\oplus$ on the paths
- $r(\text{list}) \uplus \text{list} = \text{list}$ and $r(\text{list}) \neq \text{list}$
  $\leadsto$ $\text{list} \prec_{\uplus} r(\text{list})$
  $\leadsto$ $\lhd$ strictly idempotent r-operator
- Self-stabilizing algorithm for routing tables construction in message passing environment
  with shortest path

Summary

1 Contribution
2 Generic approach with r-semi-groups
3 Main results
4 Using r-semi-groups
5 Conclusion

Contribution
Generic approach
Algorithm
Reminder
Operators
Main results
Using r-semi-groups
Method
HOWTO
Conclusion

Conclusion

- Hard to design (and prove) self-stabilizing algorithms
- r-semi-group: a generalization of the Abelian idempotent semi-group
  generic approach for designing stabilizing algorithms solving static tasks
  even for message passing unreliable networks

- Future works:
  - relation total/partial order $\leftrightarrow$ atomicity?
    currently, some restrictions if partial order
  - completeness?