Characterization of Logics on Infinite Linear Orderings

Thomas Colcombet
3.3.2015
Séminaire général du
Laboratoire d'informatique Gaspard-Monge

Linear orderings
Words
Logics

Monadic Second-Order Logic

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Words signature: binary order + predicates for each letter.

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Words signature: binary order + predicates for each letter.

In FO, « is dense »: for all $x<y$ there is some z such that $x<z<y$

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Words signature: binary order + predicates for each letter.

In FO, « is dense »: for all x<y there is some z such that x<z<y

In MSO, « is scattered »: no (induced) sub-ordering is dense

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Words signature: binary order + predicates for each letter.

In FO, « is dense »: for all x<y there is some z such that x<z<y

In MSO, « is scattered »: no (induced) sub-ordering is dense

In MSO, « is finite »: the first and last positions exist and are reachable one from the other by successor steps

Monadic Second-Order Logic

Monadic second-order logic (MSO)
- quantify over elements x,y,…
- quantify over sets of elements X,Y,… (monadic variables)
- use there relation predicates of the ambient signature
- Boolean connectives

For instance over the di-graph signature, « t is reachable from s »:
every set containing s and closed under edge relation also contains t.

Words signature: binary order + predicates for each letter.

In FO, « is dense »: for all $x<y$ there is some z such that $x<z<y$

In MSO, « is scattered »: no (induced) sub-ordering is dense

In MSO, « is finite »: the first and last positions exist and are reachable one from the other by successor steps

In MSO, « is complete »: all subsets have a supremum

History

History

Elgot - Büchi60
MSO=reg (finite words)
decidable

History

Elgot - Büchi60
MSO=reg (finite words)
decidable

[Büchi62]: ω-words
decidable

(Q,<): [Rabin69]

(Q,<): [Shelah75]
(R,<): [Shelah75] (undecidable)

MSO=recognizable [Carton,C.,Puppis]
over countable linear orderings

History

Elgot - Büchi60
MSO=reg (finite words)
decidable

[Büchi62]: ω-words
decidable

(Q,<): [Rabin69]

(Q,<): [Shelah75]
(R,<): [Shelah75] (undecidable)

MSO=recognizable [Carton,C.,Puppis]
over countable linear orderings

[Schützenberger65]
[McNaughton&Papert71]
FO-definable = aperiodic

Many logics…

History

Elgot - Büchi60
MSO=reg (finite words)
decidable

[Büchi62]: ω-words
decidable

(Q,<): [Rabin69]

(Q,<): [Shelah75]
(R,<): [Shelah75] (undecidable)

MSO=recognizable [Carton,C.,Puppis]
over countable linear orderings

?

[Schützenberger65]
[McNaughton&Papert71]
FO-definable = aperiodic

Many logics…

Linear orderings and infinite words

Linear orderings and infinite words

Linear ordering: $\alpha=(L,<)$ with $<$ total (here L is always countable)

Linear orderings and infinite words

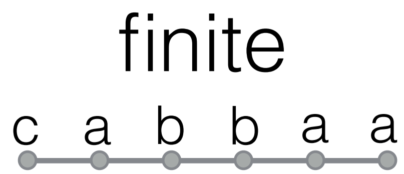
Linear ordering: $\alpha=(L,<)$ with $<$ total (here L is always countable)

(Countable) **word**: map $u : \alpha \to A$ (A alphabet)

Linear orderings and infinite words

Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

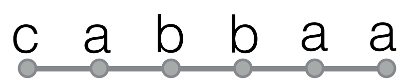


Linear orderings and infinite words

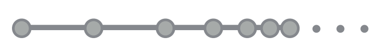
Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

finite



domain ω (N,<)

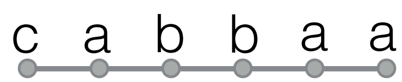


Linear orderings and infinite words

Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

finite



domain ω (N,<)

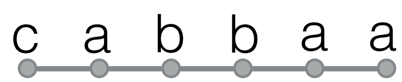
domain ω* (-N,<)

Linear orderings and infinite words

Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

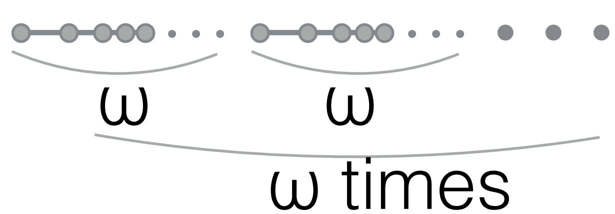
finite



domain ω (N,<)

domain ω* (-N,<)

well ordered domain (ordinal)

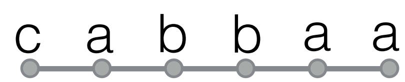


Linear orderings and infinite words

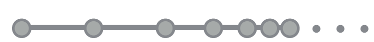
Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

finite

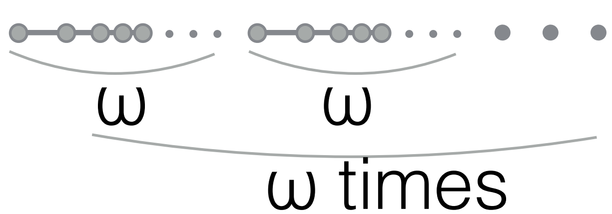


domain ω (N,<)



domain ω* (-N,<)

well ordered domain (ordinal)



scattered

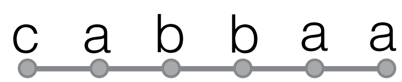
(no dense sub-ordering)

Linear orderings and infinite words

Linear ordering: $\alpha=(L,<)$ with < total (here L is always countable)

(Countable) **word**: map $u : \alpha \rightarrow A$ (A alphabet)

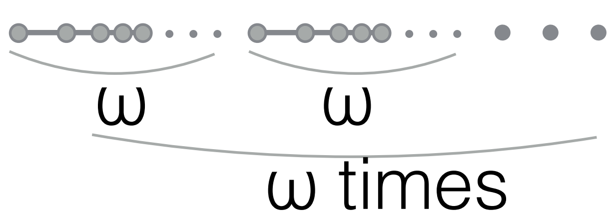
finite



domain ω (N,<)

domain ω* (-N,<)

well ordered domain (ordinal)



scattered

(no dense sub-ordering)

perfect shuffle {a,b}

domain (Q,<)

every letter appears densely

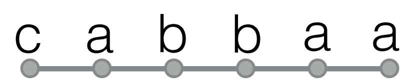
(unique up to isomorphism)

Linear orderings and infinite words

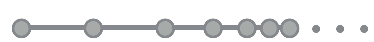
Linear ordering: $\alpha=(L,<)$ with $<$ total (here L is always countable)

(Countable) **word**: map $u : \alpha \to A$ (A alphabet)

finite

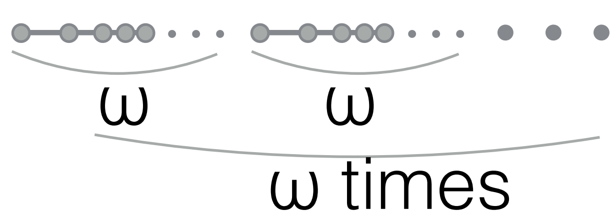


domain ω $(\mathbb{N},<)$



domain ω^* $(-\mathbb{N},<)$

well ordered domain (ordinal)



scattered

(no dense sub-ordering)

perfect shuffle $\{a,b\}$

domain $(\mathbb{Q},<)$
every letter appears densely
(unique up to isomorphism)

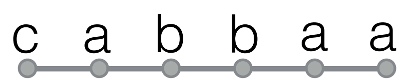
complete

Linear orderings and infinite words

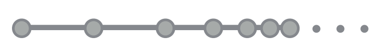
Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

finite

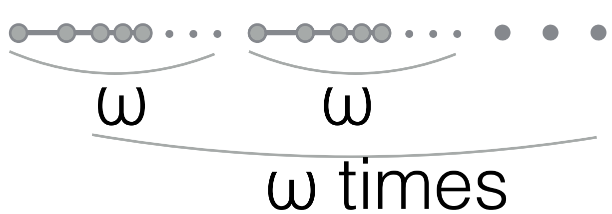


domain ω (N,<)



domain ω* (-N,<)

well ordered domain (ordinal)



scattered

(no dense sub-ordering)

perfect shuffle {a,b}

domain (Q,<)

every letter appears densely

(unique up to isomorphism)

complete

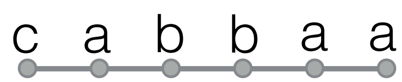
incomplete

Linear orderings and infinite words

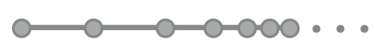
Linear ordering: α=(L,<) with < total (here L is always countable)

(Countable) **word**: map u : α→A (A alphabet)

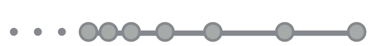
finite



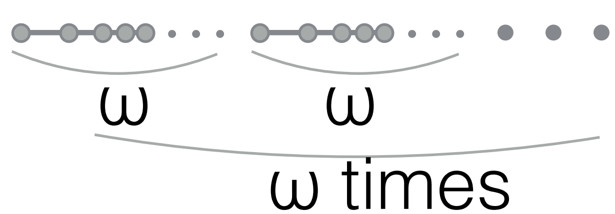
domain ω (N,<)



domain ω* (-N,<)



well ordered domain (ordinal)



scattered

(no dense sub-ordering)

perfect shuffle {a,b}

domain (Q,<)

every letter appears densely

(unique up to isomorphism)

complete

incomplete

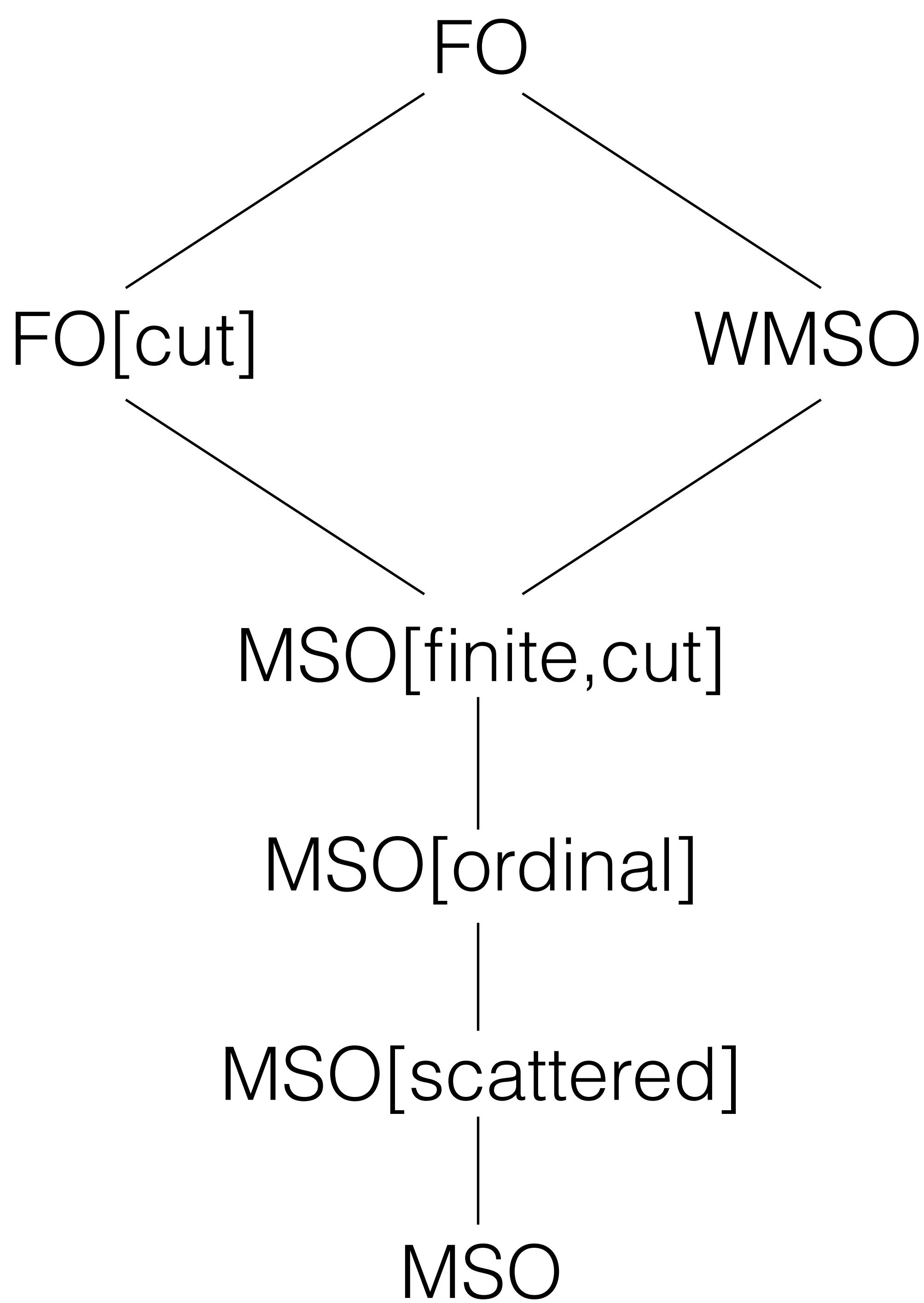
gap

= natural Dedekind cut

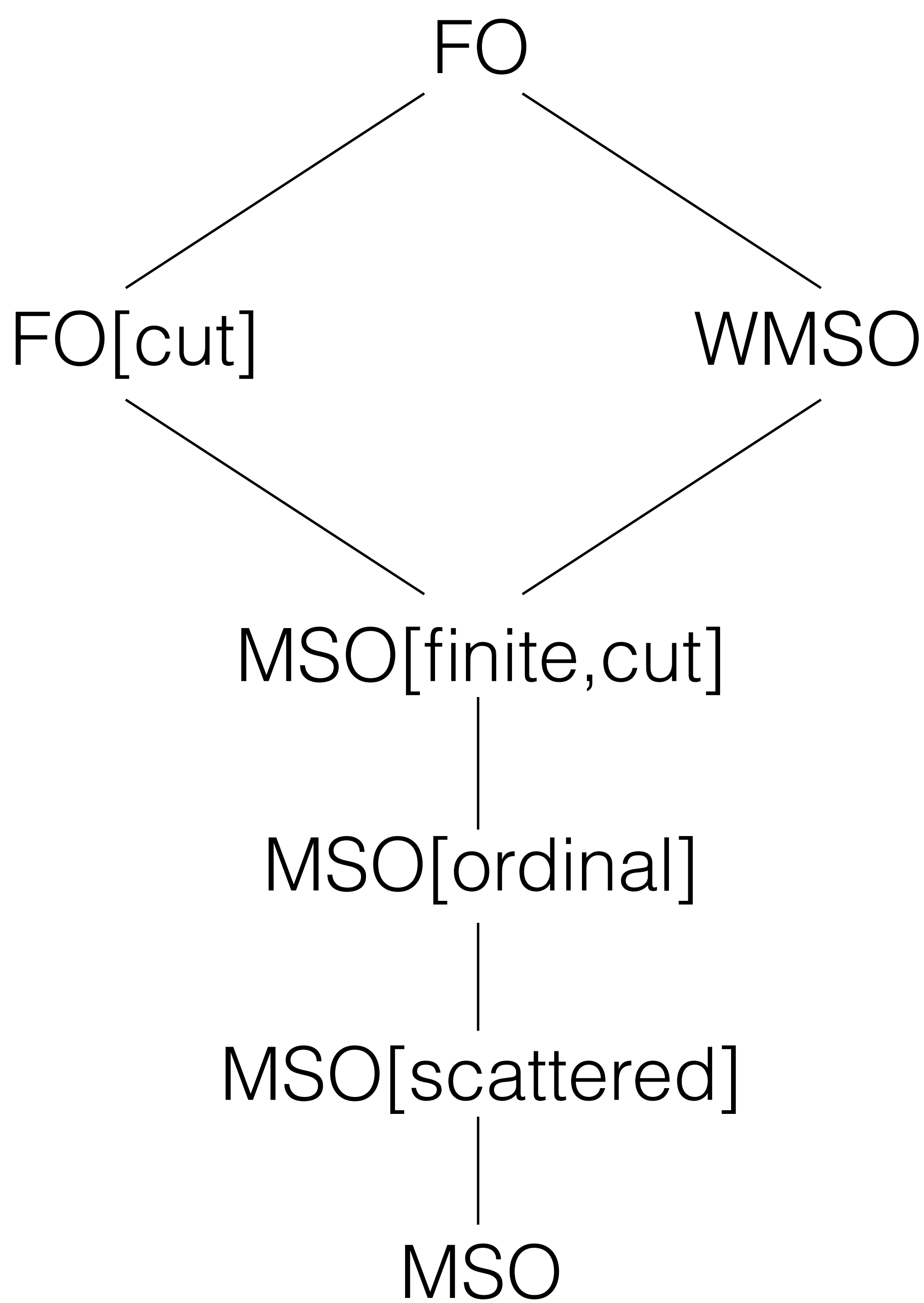
Restricting the set quantifier

Range of set quantifiers	Name of the logic
singleton sets	first-order logic (FO) « is dense », « has length k »
cuts	first-order logic with cuts (FO[cut]) « is well ordered », « is complete », « is finite »
finite sets	weak monadic second-order logic (WMSO) « is finite », « has even length »
finite sets and cuts	MSO[finite,cut] « there is an even number of gaps »
well ordered sets	MSO[ordinal] ...
scattered sets	MSO[scattered] « is scattered »
all sets	MSO « there are two sets ‘dense in each other’ »

Structure

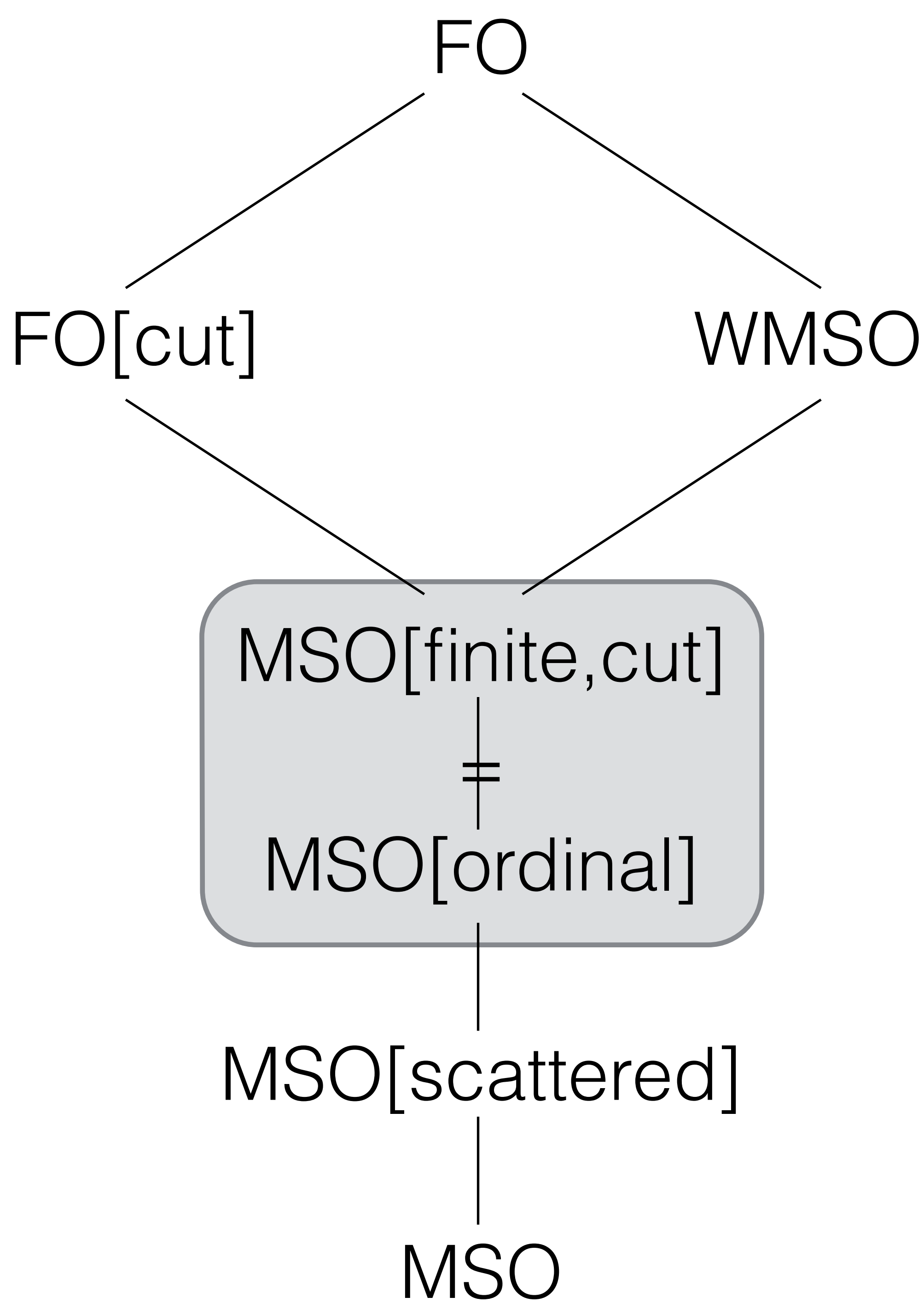


Structure



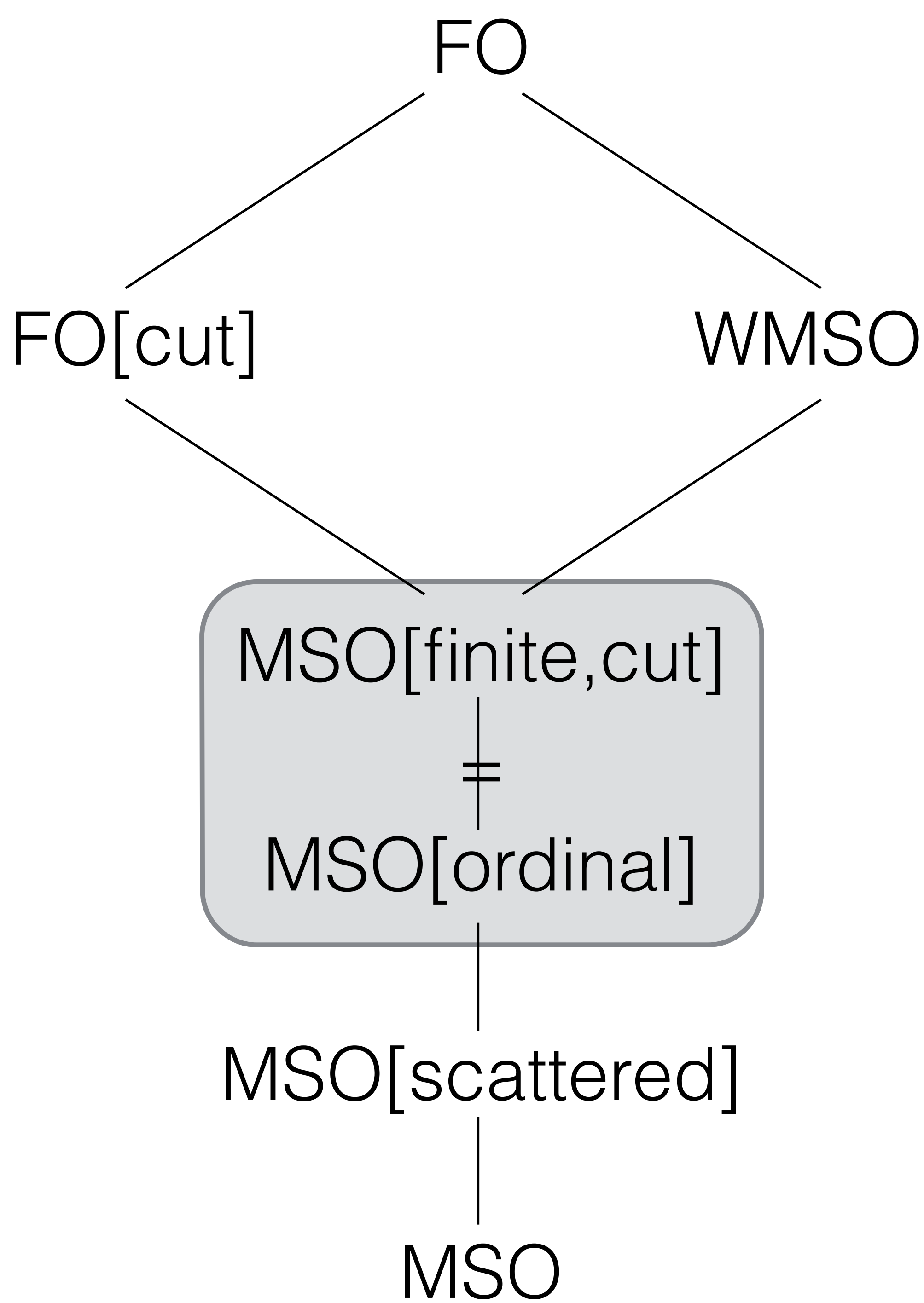
Can we separate these logics ?

Structure



Can we separate these logics ?

Structure



Can we separate these logics ?

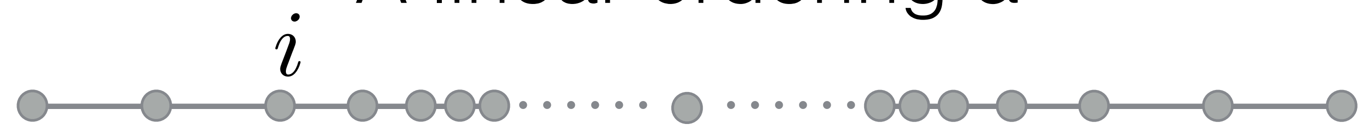
Can we characterize effectively these logics ?

An algebraic approach: $\circ$-monoid

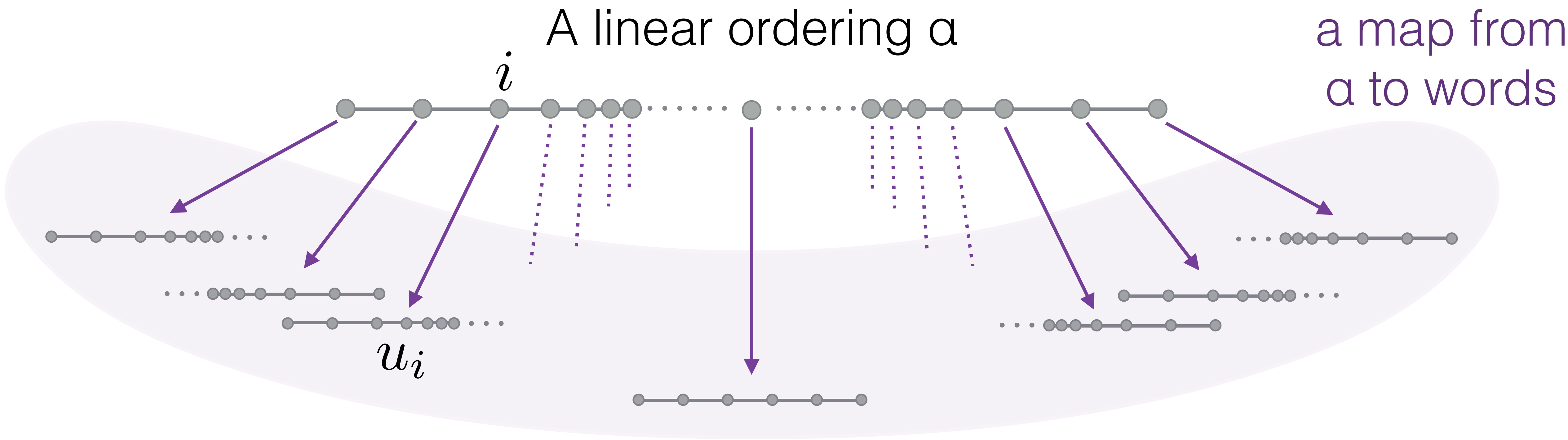
Generalized concatenation

Generalized concatenation

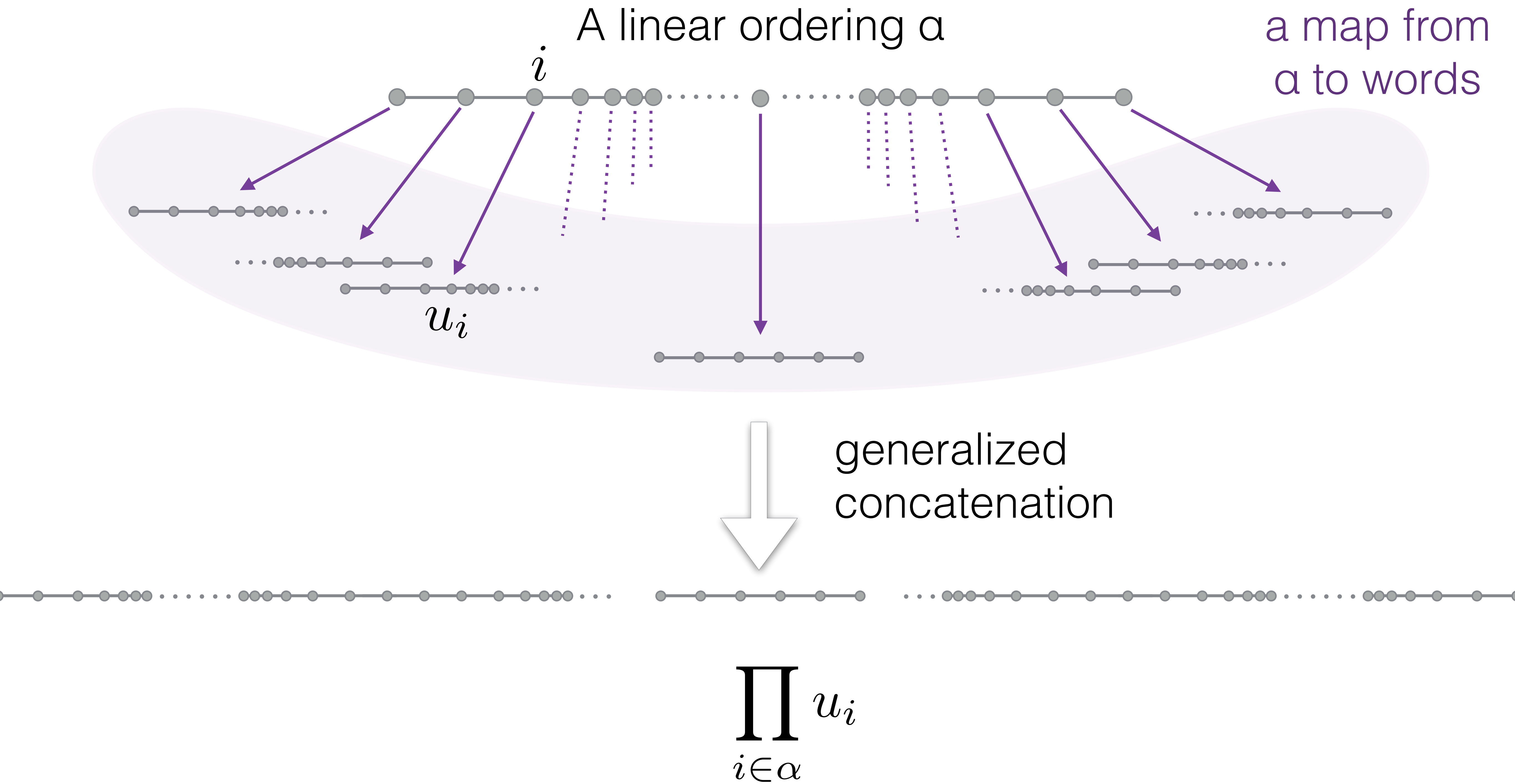
A linear ordering α



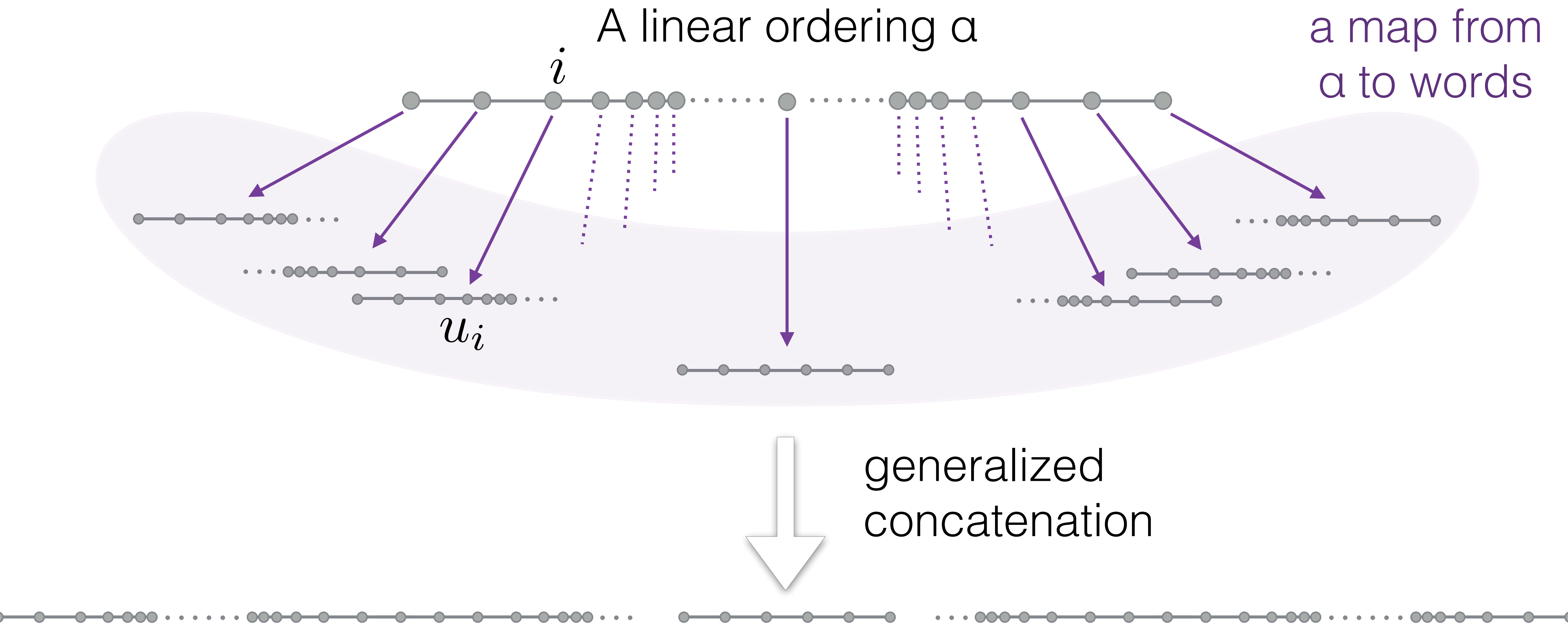
Generalized concatenation



Generalized concatenation



Generalized concatenation



$$\prod_{i \in \alpha} u_i$$

Said differently, this is a flattening operation : $\prod : (A^\circ)^\circ \to A^\circ$

$\circ$-monoids

○-monoids

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

○-monoids

$$\pi(a) = a$$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

○-monoids

$$\pi(a) = a$$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

Example: $\left(A^\circ, \prod\right)$ is the free ○-monoid generated by A.

○-monoids

$$\pi(a) = a$$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

Example: $\left(A^\circ, \prod\right)$ is the free ○-monoid generated by A.

Example:
M={1,f,0} with:

$$\pi(u) = \begin{cases} 1 & \text{if } u \text{ consists only of 1's} \\ f & \text{if } u \text{ has one but finitely many f's, and no 0} \\ 0 & \text{otherwise} \end{cases}$$

○-monoids

$$\pi(a) = a$$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

Example: $\left(A^\circ, \prod\right)$ is the free ○-monoid generated by A.

Example:
M={1,f,0} with:

$$\pi(u) = \begin{cases} 1 & \text{if } u \text{ consists only of 1's} \\ f & \text{if } u \text{ has one but finitely many f's, and no 0} \\ 0 & \text{otherwise} \end{cases}$$

A morphism of ○-monoid h is such that $h\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} h(u_i)\right)$

○-monoids

$$\pi(a) = a$$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

Example: $\left(A^\circ, \prod\right)$ is the free ○-monoid generated by A.

Example:
M={1,f,0} with: $$\pi(u) = \begin{cases} 1 & \text{if } u \text{ consists only of 1's} \\ f & \text{if } u \text{ has one but finitely many f's, and no 0} \\ 0 & \text{otherwise} \end{cases}$$

A morphism of ○-monoid h is such that $h\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} h(u_i)\right)$

Given a finite monoid M, a ○-morphism h from A° to M, and F⊆M, M,h,F recognizes $\{u \in A^\circ \ : \ h(u) \in F\}$

○-monoids

$\pi(a) = a$

A ○-monoid (M,$\boldsymbol{\pi}$) is a set M equipped with a product $\boldsymbol{\pi}$: M° → M that satisfies generalized associativity:

$$\pi\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} \pi(u_i)\right)$$

Example: $\left(A^\circ, \prod\right)$ is the free ○-monoid generated by A.

Example:
M={1,f,0} with:

$$\pi(u) = \begin{cases} 1 & \text{if } u \text{ consists only of 1's} \\ f & \text{if } u \text{ has one but finitely many f's, and no 0} \\ 0 & \text{otherwise} \end{cases}$$

A morphism of ○-monoid h is such that $h\left(\prod_{i\in\alpha} u_i\right) = \pi\left(\prod_{i\in\alpha} h(u_i)\right)$

Given a finite monoid M, a ○-morphism h from A° to M, and F⊆M, M,h,F recognizes $\{u \in A^\circ \ : \ h(u) \in F\}$

Example:
with F={1,f}

$$h(u) = \begin{cases} 1 & \text{if } u \text{ has no } a\text{'s} \\ f & \text{if } u \text{ has finitely many } a\text{'s} \\ 0 & \text{ortherwise} \end{cases}$$

M,h,F recognize « finitely many a's »

Recognizability = definability

Recognizability = definability

Schützenberger-Elgot-Büchi: A language of finite words is definable in monadic second-order logic if and only if it is recognizable by a finite monoid.

Furthermore, there is a minimal such monoid: the syntactic monoid.

Recognizability = definability

Schützenberger-Elgot-Büchi: A language of finite words is definable in monadic second-order logic if and only if it is recognizable by a finite monoid.

Furthermore, there is a minimal such monoid: the syntactic monoid.

Theorem [Shelah75 & CCP11]: A language of countable words is definable if and only if it is recognizable by a finite $\circ$-monoid.

Furthermore there is a syntactic $\circ$-monoid.

Furthermore, finite $\circ$-monoids can be effectively handled.

Effectiveness: induced operations

Effectiveness: induced operations

Unit: M

$1 = \pi(\varepsilon)$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

ω-iteration: M→M

$$a^\omega = \pi(\underbrace{aaa\ldots}_{\omega})$$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

ω-iteration: M→M

$$a^\omega = \pi(\underbrace{aaa\ldots}_{\omega})$$

ω*-iteration

$$a^\omega = \pi(\underbrace{\ldots aaa}_{\omega *})$$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

shuffle η: $\mathcal{P}$(M)→M

$$\{a, b\}^\eta = \pi(perfectshuffle(a, b))$$

ω-iteration: M→M

$$a^\omega = \pi(\underbrace{aaa\ldots}_{\omega})$$

ω*-iteration

$$a^\omega = \pi(\underbrace{\ldots aaa}_{\omega *})$$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

shuffle η: $\mathcal{P}$(M)→M

$$\{a, b\}^{\eta} = \pi(perfectshuffle(a, b))$$

domain (Q,<)
every letter appears densely
(unique up to isomorphism)

ω-iteration: M→M

$$a^{\omega} = \pi(\underbrace{aaa\ldots}_{\omega})$$

ω*-iteration

$$a^{\omega} = \pi(\underbrace{\ldots aaa}_{\omega*})$$

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

shuffle η: $\mathcal{P}$(M)→M

$$\{a, b\}^{\eta} = \pi(\mathit{perfectshuffle}(a, b))$$

a b a b a b

domain (Q,<)
every letter appears densely
(unique up to isomorphism)

ω-iteration: M→M

$$a^{\omega} = \pi(\underbrace{aaa\ldots}_{\omega})$$

ω*-iteration

$$a^{\omega} = \pi(\underbrace{\ldots aaa}_{\omega *})$$

Theorem[CCP11]: There are equalities (A) such that:

every operations induced by a product satisfy equalities (A),

and

given 1,·,ω,ω*,η over some finite M satisfying these equalities, there is a product π inducting them.

Effectiveness: induced operations

Unit: M

$$1 = \pi(\varepsilon)$$

Binary product: M×M→M

$$a \cdot b = \pi(ab)$$

shuffle η: $\mathcal{P}$(M)→M

$$\{a, b\}^\eta = \pi(\mathit{perfectshuffle}(a, b))$$

a b a b a b

domain (Q,<)
every letter appears densely
(unique up to isomorphism)

ω-iteration: M→M

$$a^\omega = \pi(\underbrace{aaa\ldots}_{\omega})$$

ω*-iteration

$$a^\omega = \pi(\underbrace{\ldots aaa}_{\omega *})$$

Theorem[CCP11]: There are equalities (A) such that:

every operations induced by a product satisfy equalities (A),

and

given 1,·,ω,ω*,η over some finite M satisfying these equalities, there is a product π inducting them.

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

$$(a^n)^\omega = a^\omega$$

$$(a \cdot b)^\omega = a \cdot (b \cdot a)^\omega$$

$$\{a\}^\eta = \{a\}^\eta \cdot a \cdot \{a\}^\eta$$

$$\vdots$$

Examples

Examples

« finitely many a's »

	1	f	0
1	1	f	0
f	f	f	0
0	0	0	0

	1	f	0
ω	1	0	0

	1	f	0
ω*	1	0	0

	{1}	{f,*},{0,*}
η	1	0

h(a)=f
f(b)=1

F={1,f}

Examples

« finitely many a's »

	1	f	0
1	1	f	0
f	f	f	0
0	0	0	0

	1	f	0
ω	1	0	0

	1	f	0
ω*	1	0	0

	{1}	{f,*},{0,*}
η	1	0

h(a)=f
f(b)=1

F={1,f}

« a's are left-closed »

	1	a	b	m	0
1	1	a	b	m	0
a	a	a	m	m	0
b	b	0	b	0	0
m	m	0	m	0	0
0	0	0	0	0	0

	1	a	b	m	0
ω	1	a	b	0	0

	1	a	b	m	0
ω*	1	a	b	0	0

a = « …aaa… »
b = « …bbb… »
m = « …aaa…bbb… »
0 = « *b*a* »

Characterizing logics

First order cannot detect gaps…

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

Theorem [Bès&Carton13]: A language of countable scattered words is definable in FO if and only if every idempotent is gap insensitive.

$$e \cdot e = e \qquad\qquad e^{\omega} \cdot e^{\omega *} = e$$

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

Theorem [Bès&Carton13]: A language of countable scattered words is definable in FO if and only if every idempotent is gap insensitive.

$e \cdot e = e$ $\qquad$ $e^{\omega} \cdot e^{\omega *} = e$

 « looks as »  when sufficiently long.

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

Theorem [Bès&Carton13]: A language of countable scattered words is definable in FO if and only if every idempotent is gap insensitive.

$$e \cdot e = e \qquad e^{\omega} \cdot e^{\omega *} = e$$

 « looks as »  when sufficiently long.

Remark: « All idempotents are gap insensitive » implies aperiodicity.

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

Theorem [Bès&Carton13]: A language of countable scattered words is definable in FO if and only if every idempotent is gap insensitive.

$$e \cdot e = e \qquad\qquad e^{\omega} \cdot e^{\omega *} = e$$

 « looks as »  when sufficiently long.

Remark: « All idempotents are gap insensitive » implies aperiodicity.

Proof: Take n such that a^n is idempotent.

$$a^n = (a^n)^{\omega} \cdot (a^n)^{\omega *} = a \cdot (a^n)^{\omega} \cdot (a^n)^{\omega *} = a^{n+1}$$

First order cannot detect gaps…

Theorem[Schützenberger65,McNauthon&Papert71]: A language of finite words is definable in FO if and only if it is aperiodic.

Theorem [Bès&Carton13]: A language of countable scattered words is definable in FO if and only if every idempotent is gap insensitive.

$$e \cdot e = e \qquad\qquad e^{\omega} \cdot e^{\omega *} = e$$

 « looks as »  when sufficiently long.

Remark: « All idempotents are gap insensitive » implies aperiodicity.

Proof: Take n such that a^n is idempotent.

$$a^n = (a^n)^{\omega} \cdot (a^n)^{\omega *} = a \cdot (a^n)^{\omega} \cdot (a^n)^{\omega *} = a^{n+1}$$

Remark: The equation remains true but is not sufficient in general.

Weak monadic logic cannot detect gaps… when in an infinite situation

Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad\qquad$ $e^{\omega *} = e$

Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^{\omega} = e$ $\qquad\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

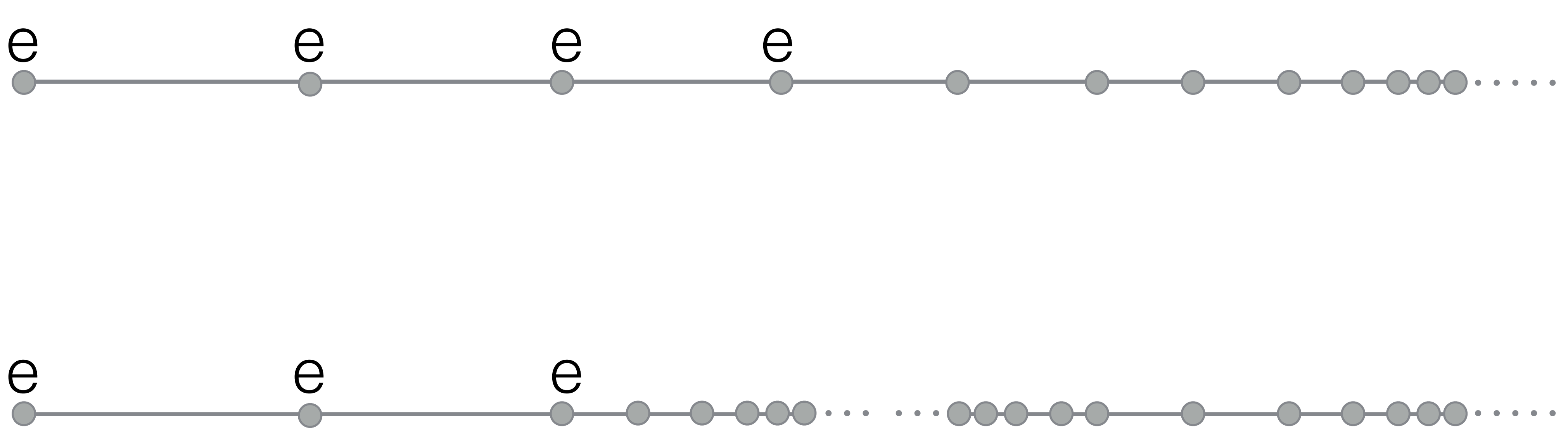
Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^{\omega} = e$ $\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

e



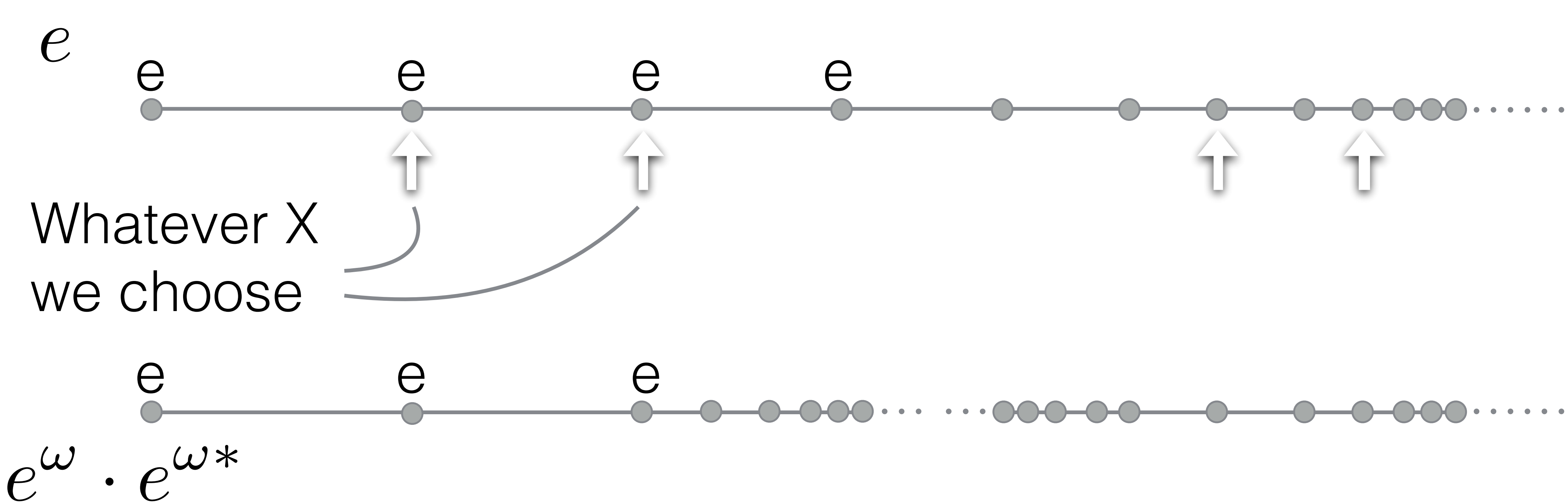
$e^{\omega} \cdot e^{\omega *}$

Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

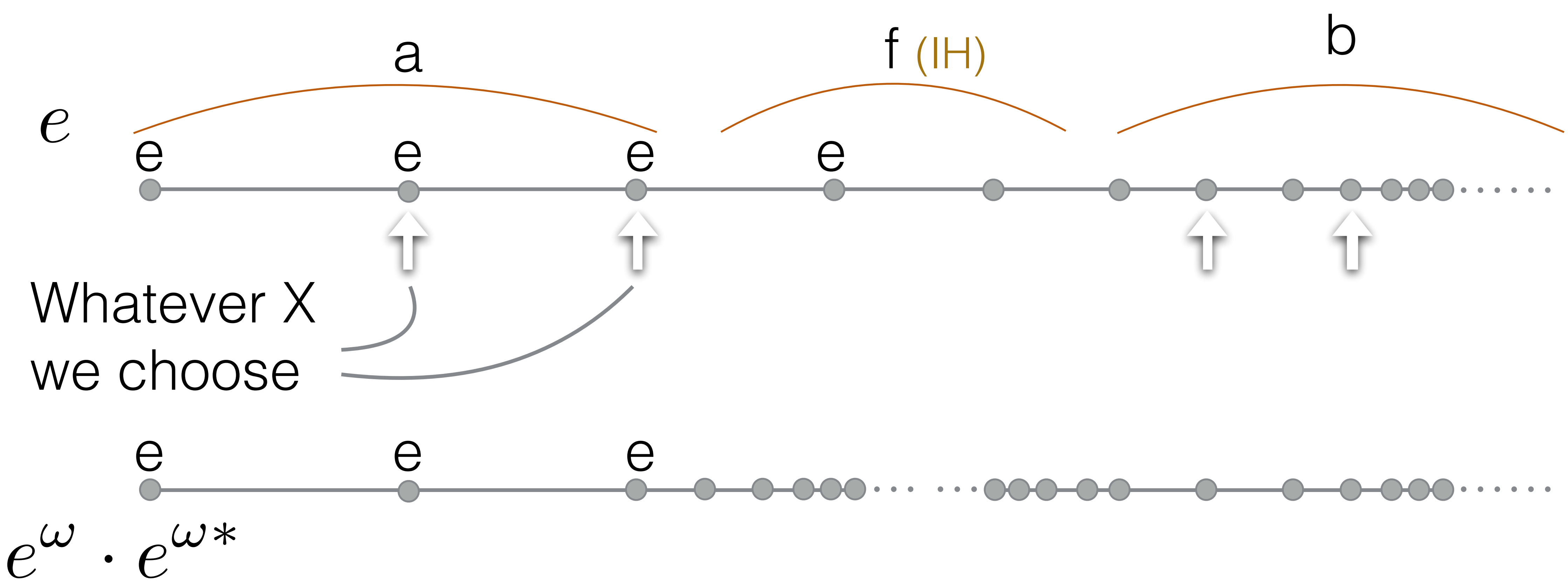


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$$e^\omega = e \qquad\qquad e^{\omega *} = e$$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

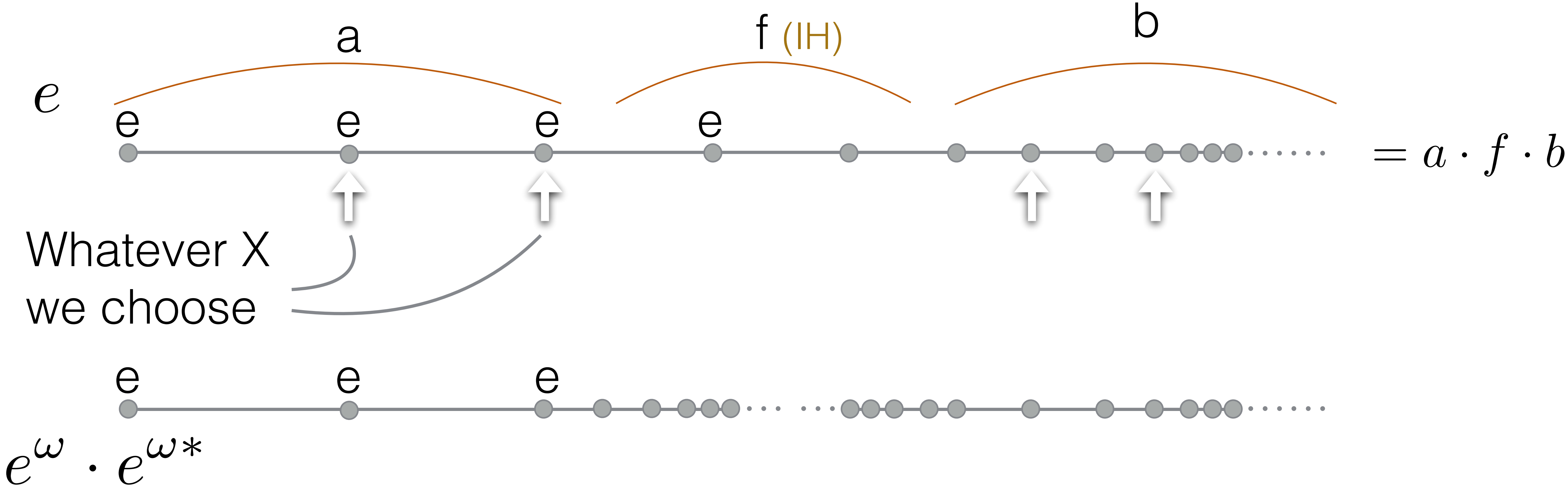


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

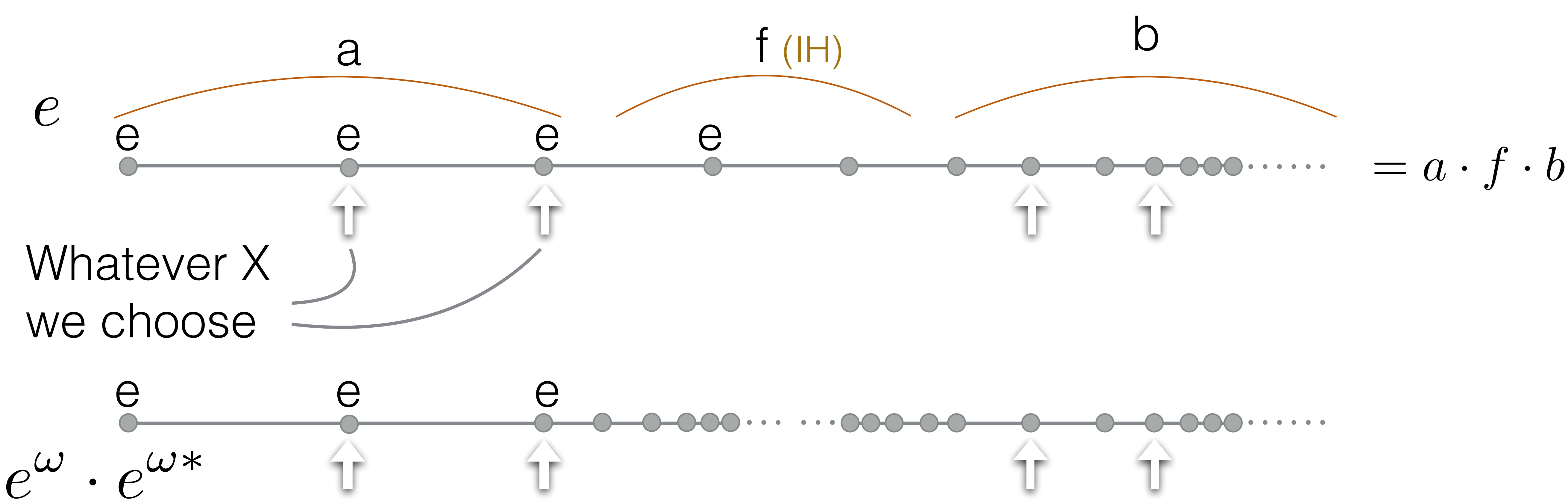


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$$e^{\omega} = e \qquad\qquad e^{\omega *} = e$$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

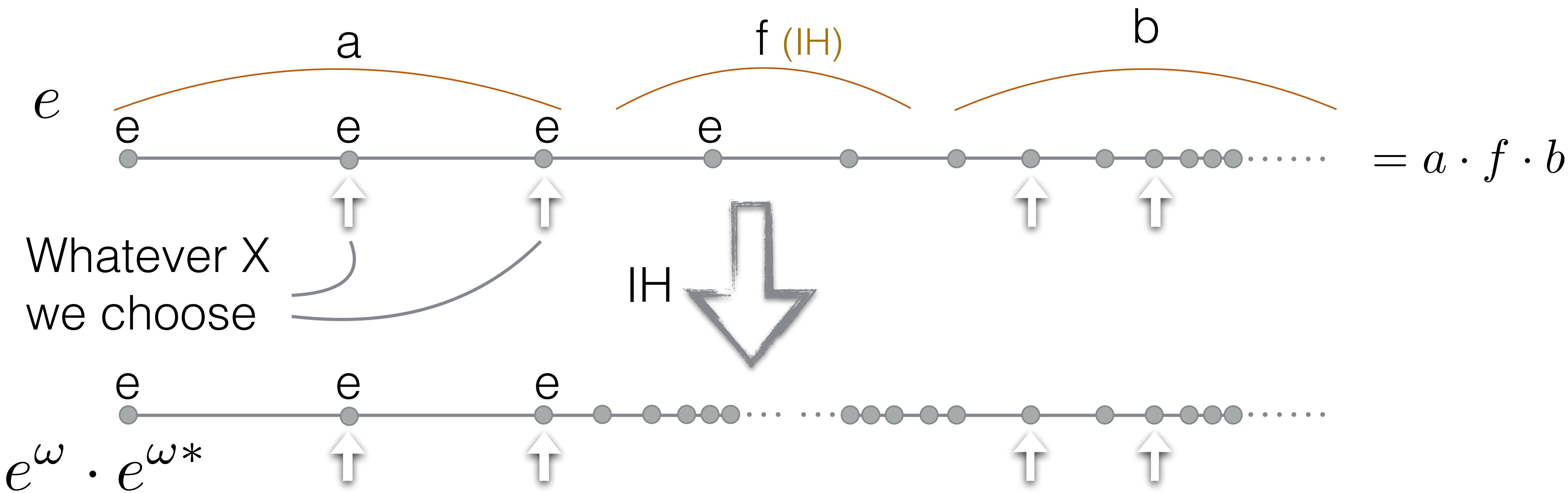


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

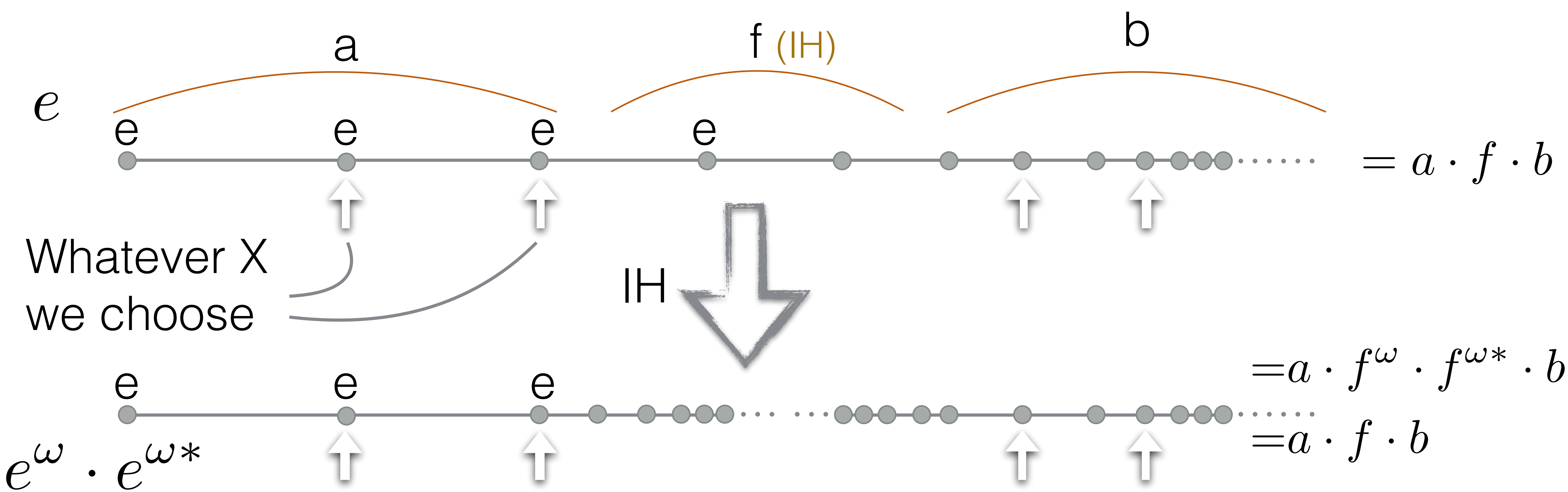


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad\qquad$ $e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.

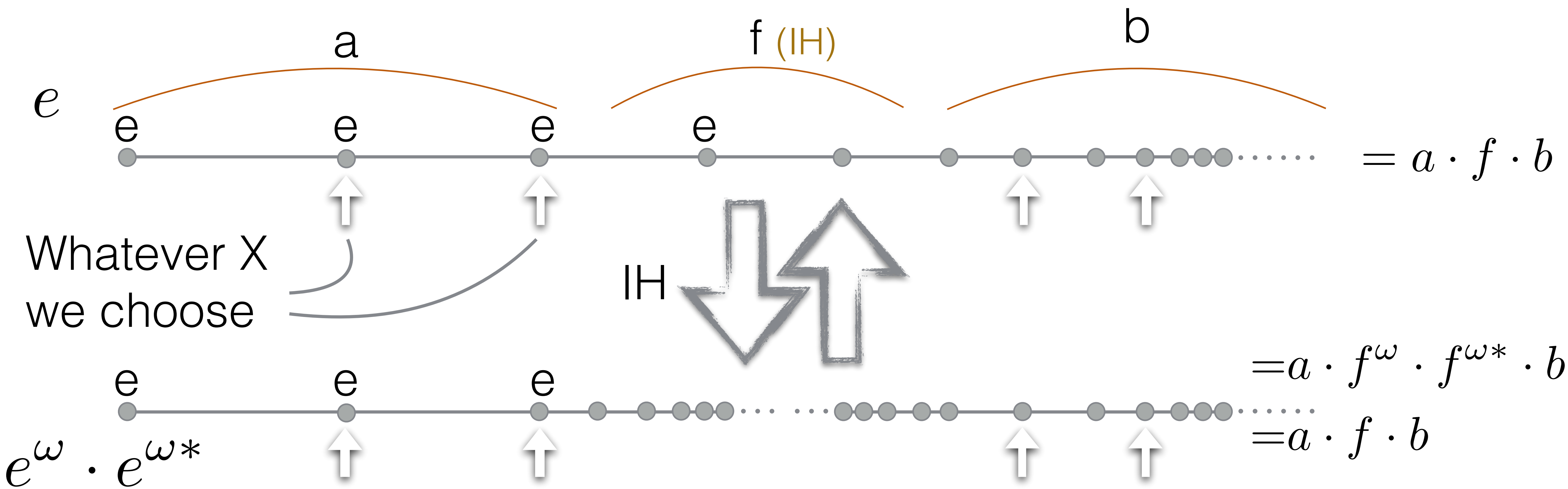


Weak monadic logic cannot detect gaps… when in an infinite situation

[Bès&Carton]: A language of scattered words is definable in WMSO if and only if all ordinal idempotents and every ordinal* idempotents are gap insensitive.

$e^\omega = e$ $\qquad e^{\omega *} = e$

IH: Assume « ϕ(X) » recognized by a monoid satisfying the property.



MSO[ordinal] cannot see scattered set

Lemma[C.&Sreejith A.V.]: Every formula of MSO[ordinal] has a syntactic ○-monoid such that every scattered idempotent is a shuffle idempotent.

$$e = e^{\omega} = e^{\omega *} \qquad e = \{e\}^{\eta}$$

MSO[ordinal] cannot see scattered set

Lemma[C.&Sreejith A.V.]: Every formula of MSO[ordinal] has a syntactic ○-monoid such that every scattered idempotent is a shuffle idempotent.

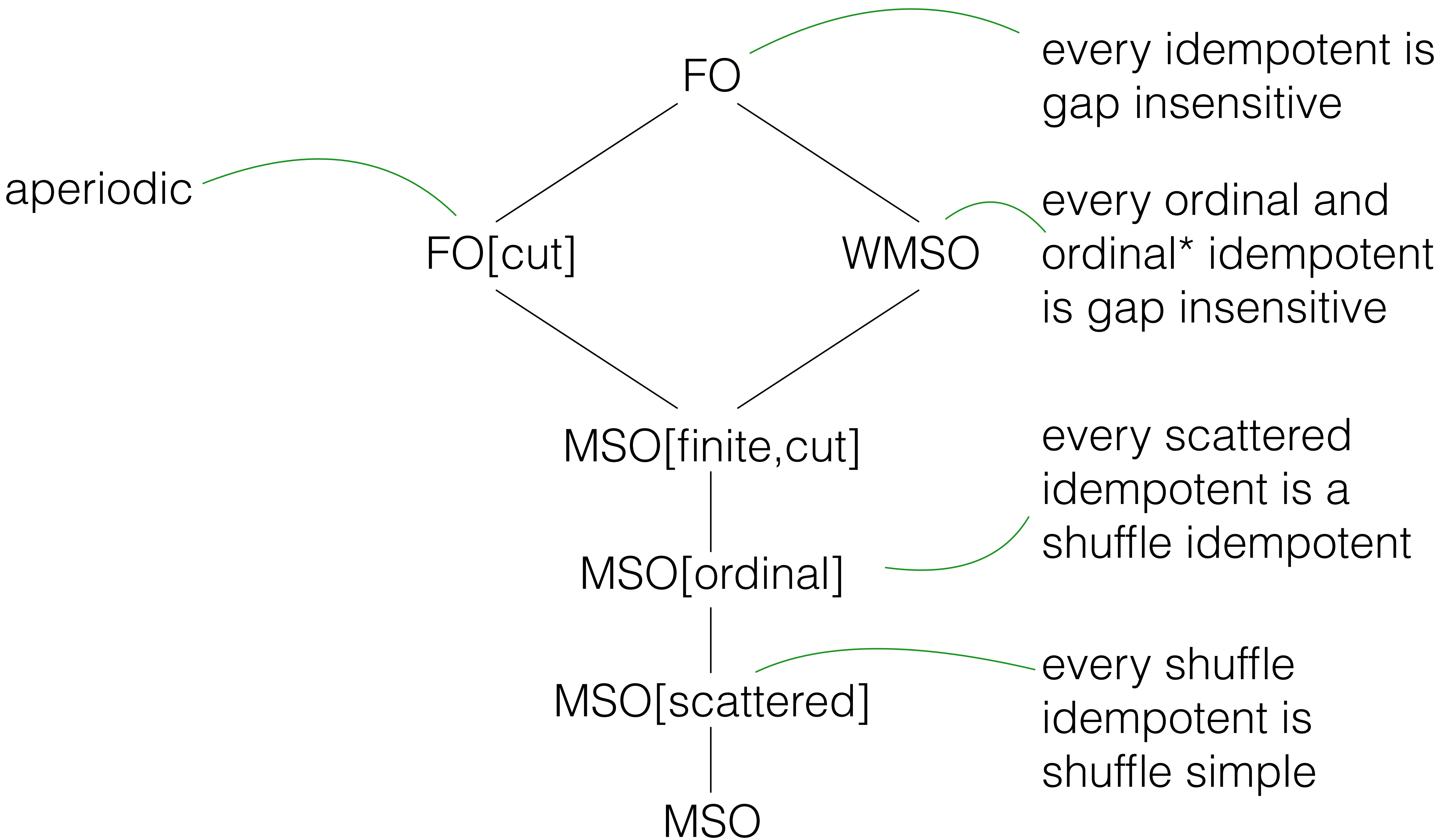
$$e = e^{\omega} = e^{\omega *} \qquad e = \{e\}^{\eta}$$

MSO[scattered]

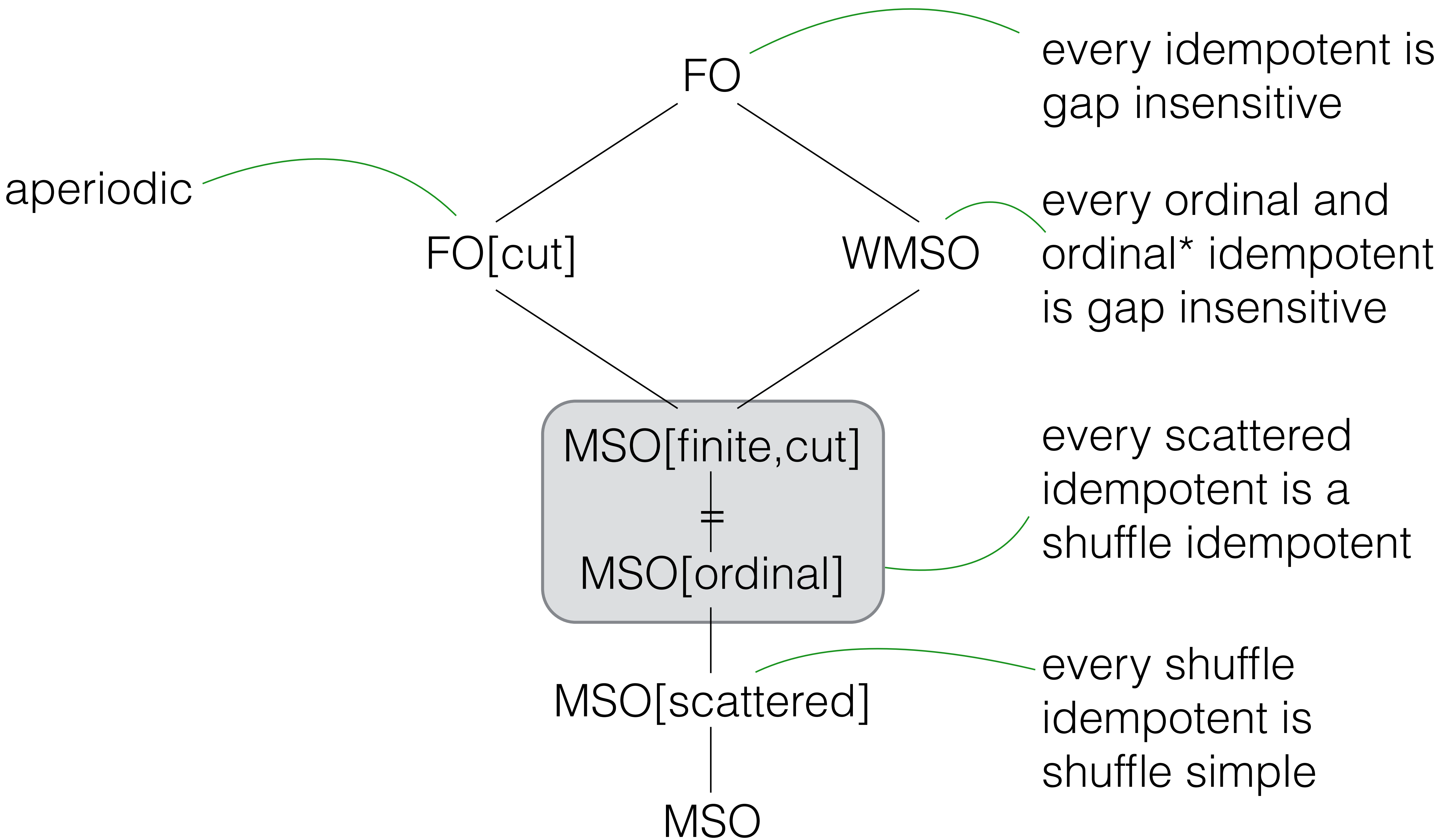
Lemma[C.&Sreejith A.V.]: Every formula of MSO[ordinal] has a syntactic ○-monoid such that every shuffle idempotent is shuffle simple.

For all K such that $e = K^{\eta}$,
and a such that $e \cdot a \cdot e = e$,
$(K \cup \{a\})^{\eta} = e$.

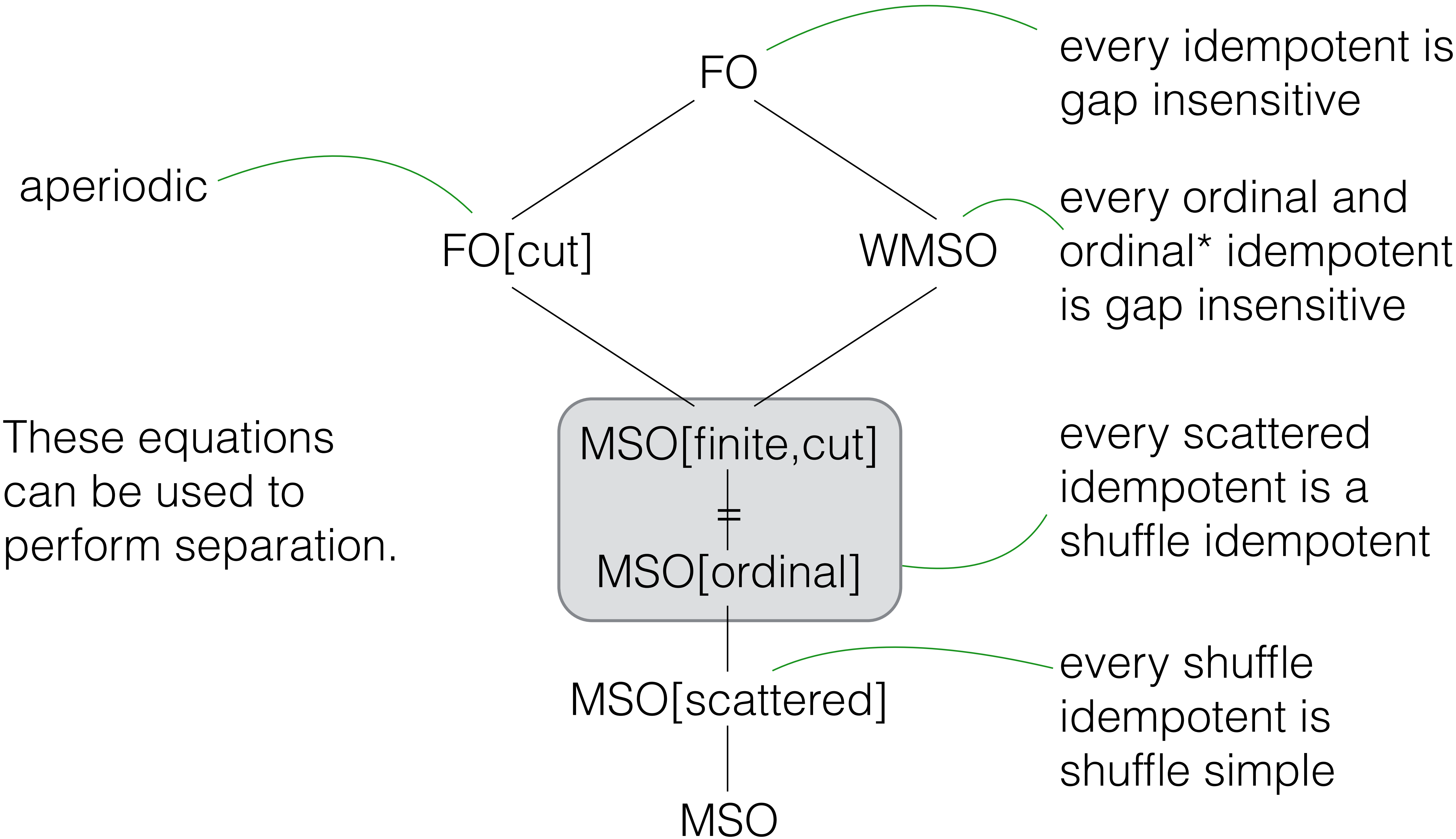
The picture


FO
every idempotent is gap insensitive
aperiodic
FO[cut]
WMSO
every ordinal and ordinal* idempotent is gap insensitive
MSO[finite,cut]
every scattered idempotent is a shuffle idempotent
MSO[ordinal]
MSO[scattered]
every shuffle idempotent is shuffle simple
MSO

The picture


FO
every idempotent is gap insensitive
aperiodic
FO[cut]
WMSO
every ordinal and ordinal* idempotent is gap insensitive
MSO[finite,cut]
=
MSO[ordinal]
every scattered idempotent is a shuffle idempotent
MSO[scattered]
every shuffle idempotent is shuffle simple
MSO

The picture



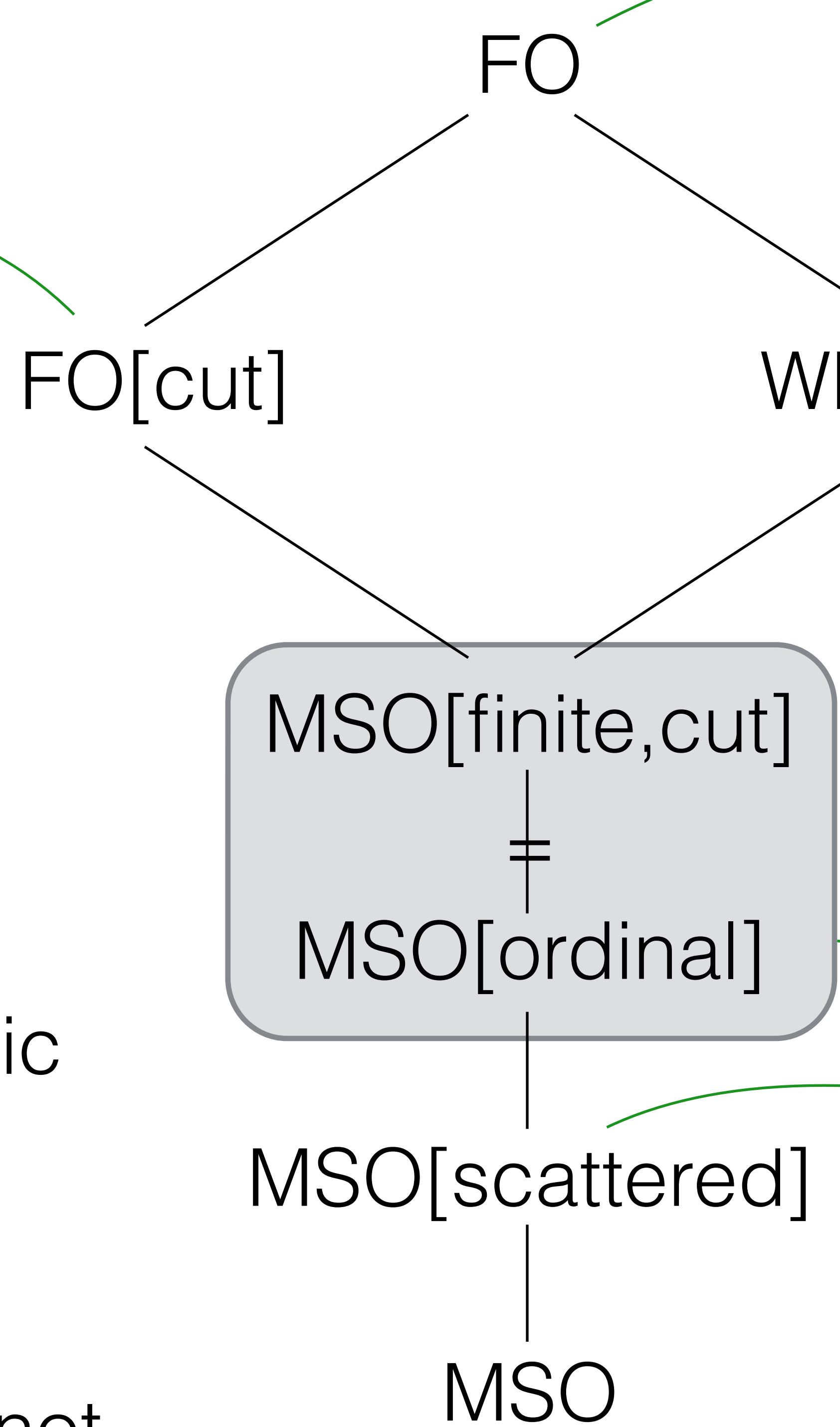
These equations can be used to perform separation.

The picture

every idempotent is gap insensitive

aperiodic

every ordinal and ordinal* idempotent is gap insensitive



These equations can be used to perform separation.

every scattered idempotent is a shuffle idempotent

Example: the syntactic ○-monoid of

« is scattered »

contains a scattered idempotent which is not a shuffle idempotent.

every shuffle idempotent is shuffle simple

Results

[C.&Sreejith A.V.]: The following properties characterize the logics: (and these logics can be separated)

	FO	FO[cut]	WMSO	MSO[finite,cut] =MSO[ordinal]	MSO[scattered]
Every idempotent is gap insensitive	✔				
Aperiodicity	(✔)	✔			
Every ordinal or ordinal* idempotent is gap insensitive	(✔)		✔		
Every scattered idempotent is a shuffle idempotent	✔	✔	✔	✔	
Every shuffle idempotent is shuffle simple	✔	✔	✔	✔	✔

To be continued…