Partially-commutative context-free processes

Wojciech Czerwiński
Sibylle Froeschle
Sławomir Lasota

Concur 2009

Outline

Outline

- What is "partially-commutative context-free" ?

Outline

- What is “partially-commutative context-free” ?
- What is “processes“ ?

Outline

- What is "partially-commutative context-free" ?
- What is "processes" ?
- Strong bisimilarity checking

Outline

- What is "partially-commutative context-free" ?
- What is "processes" ?
- Strong bisimilarity checking
- Our contribution

Outline

- What is "partially-commutative context-free" ?
- What is "processes" ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm

Outline

- What is "partially-commutative context-free" ?
- What is "processes" ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

Outline

- What is “partially-commutative context-free” ?
- What is “processes“ ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

context-free grammars

$$X \longrightarrow aXBC$$

context-free grammars

in Greibach Normal Form
under left-most derivations

$X \longrightarrow aXBC$

context-free grammars

in Greibach Normal Form
under left-most derivations

$X \xrightarrow{a} XBC$

context-free grammars

in Greibach Normal Form

under left-most derivations

$X \xrightarrow{a} XBC$ $\qquad B \xrightarrow{b}$

$X \xrightarrow{a} BC$ $\qquad C \xrightarrow{c}$

language = a ... a bc ... bc

Commutative context-free grammars

in Greibach Normal Form

under left-most derivations

$X \xrightarrow{a} XBC \qquad B \xrightarrow{b}$

$X \xrightarrow{a} BC \qquad C \xrightarrow{c}$

language = a … a bc … bc

Commutative context-free grammars

X B and C pairwise independent

$X \xrightarrow{a} XBC$ $\qquad$ $B \xrightarrow{b}$

$X \xrightarrow{a} BC$ $\qquad$ $C \xrightarrow{c}$

language =

Commutative context-free grammars

X B and C pairwise independent

$X \xrightarrow{a} XBC$ $\qquad B \xrightarrow{b}$

$X \xrightarrow{a} BC$ $\qquad C \xrightarrow{c}$

language = #a = #b = #c,

a "preceeds" b and c

Partially-commutative context-free grammars

X B and C pairwise independent

$X \xrightarrow{a} XBC$ $\qquad$ $B \xrightarrow{b}$

$X \xrightarrow{a} BC$ $\qquad$ $C \xrightarrow{c}$

language = #a = #b = #c,
a "preceeds" b and c

Partially-commutative context-free grammars

only B and C independent

$X \xrightarrow{a} XBC$

$X \xrightarrow{a} BC$

$B \xrightarrow{b}$

$C \xrightarrow{c}$

language =

Partially-commutative context-free grammars

only B and C independent

$X \xrightarrow{a} XBC$ $\qquad$ $B \xrightarrow{b}$

$X \xrightarrow{a} BC$ $\qquad$ $C \xrightarrow{c}$

language = a .. a (b .. b | c .. c)

Independence

Independence

- V = non-terminal symbols $\qquad V = \{ X, B, C \}$

Independence

- V = non-terminal symbols $\quad V = \{ X, B, C \}$
- independence I = binary symmetric and irreflexive relation on V $\quad I = \{ (B,C) \}$

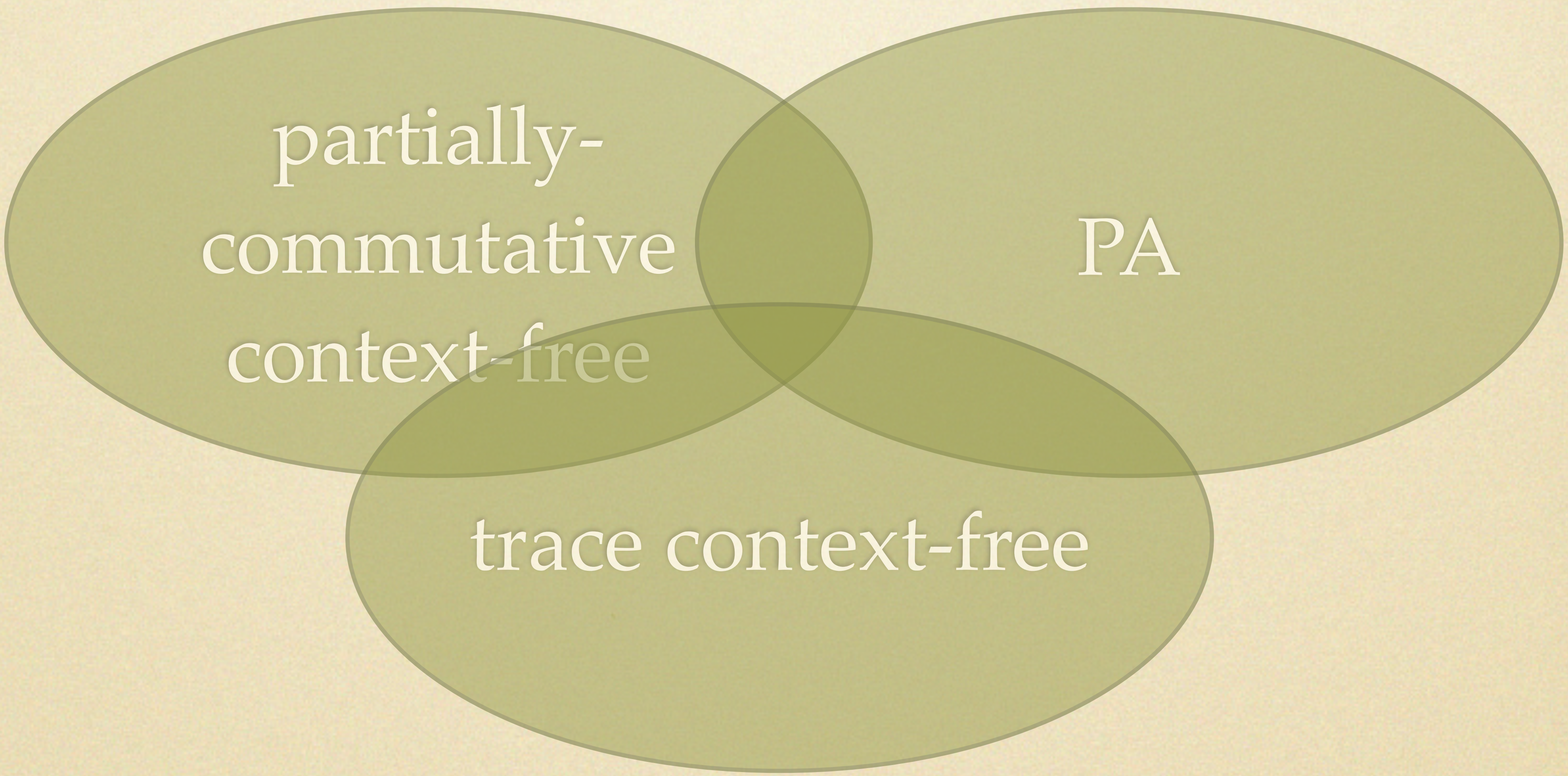
Independence

- V = non-terminal symbols $\quad V = \{ X, B, C \}$
- independence I = binary symmetric and irreflexive relation on V $\quad I = \{ (B,C) \}$
- context-free: I is identity

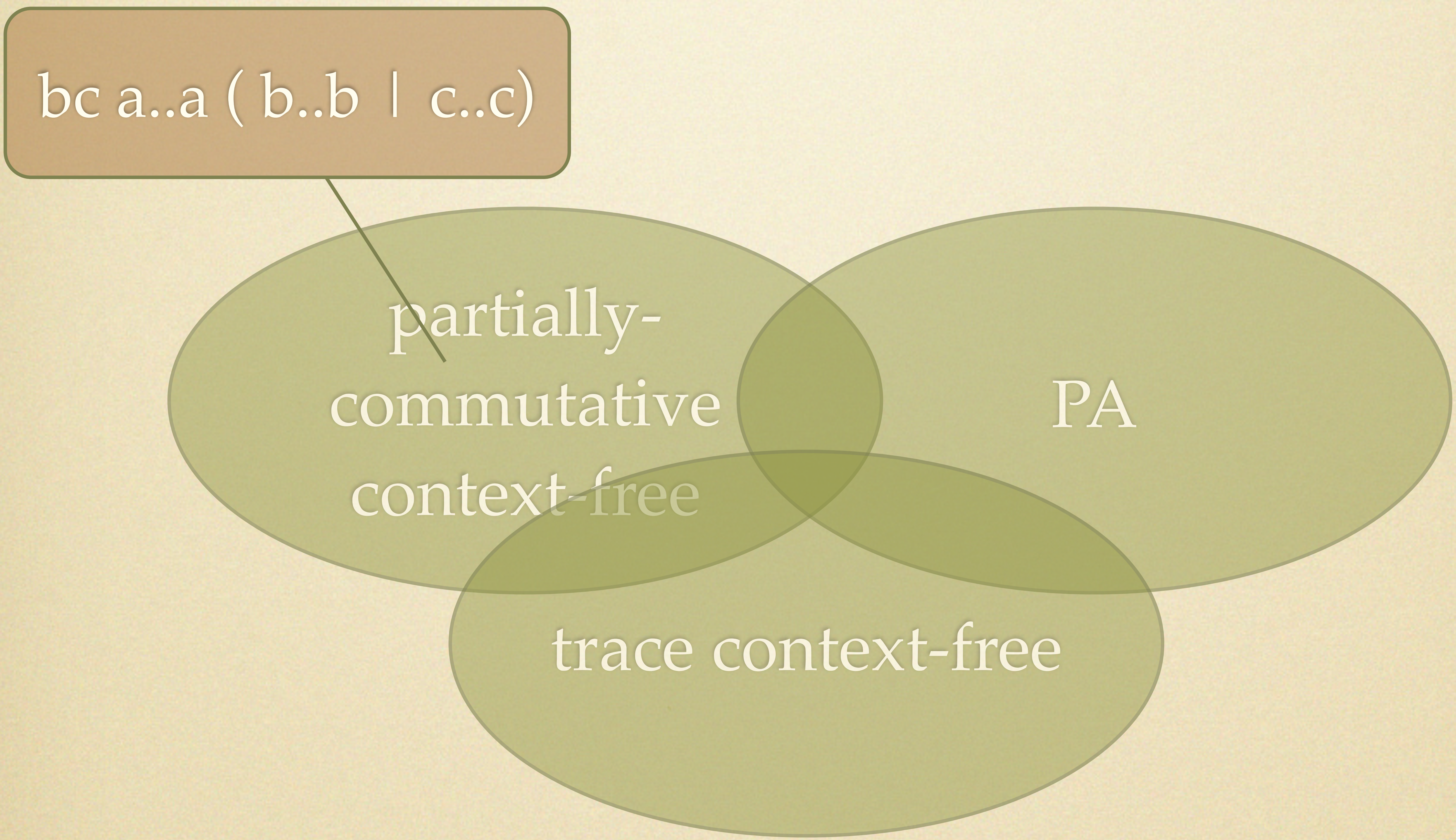
Independence

- V = non-terminal symbols $\qquad V = \{ X, B, C \}$
- independence I = binary symmetric and irreflexive relation on V $\qquad I = \{ (B,C) \}$
- context-free: I is identity
- commutative context-free: $I = V^2$

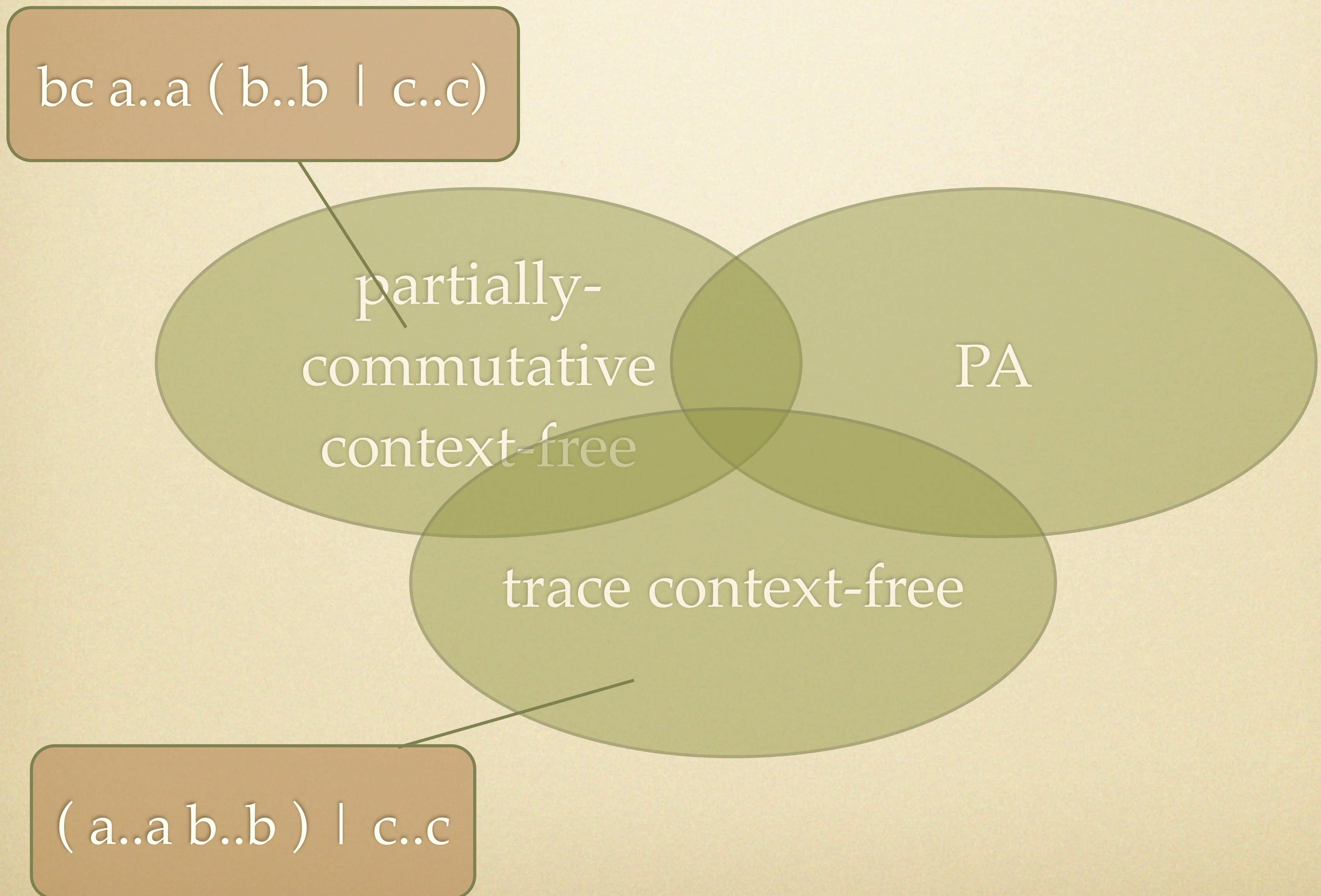
Expressibility



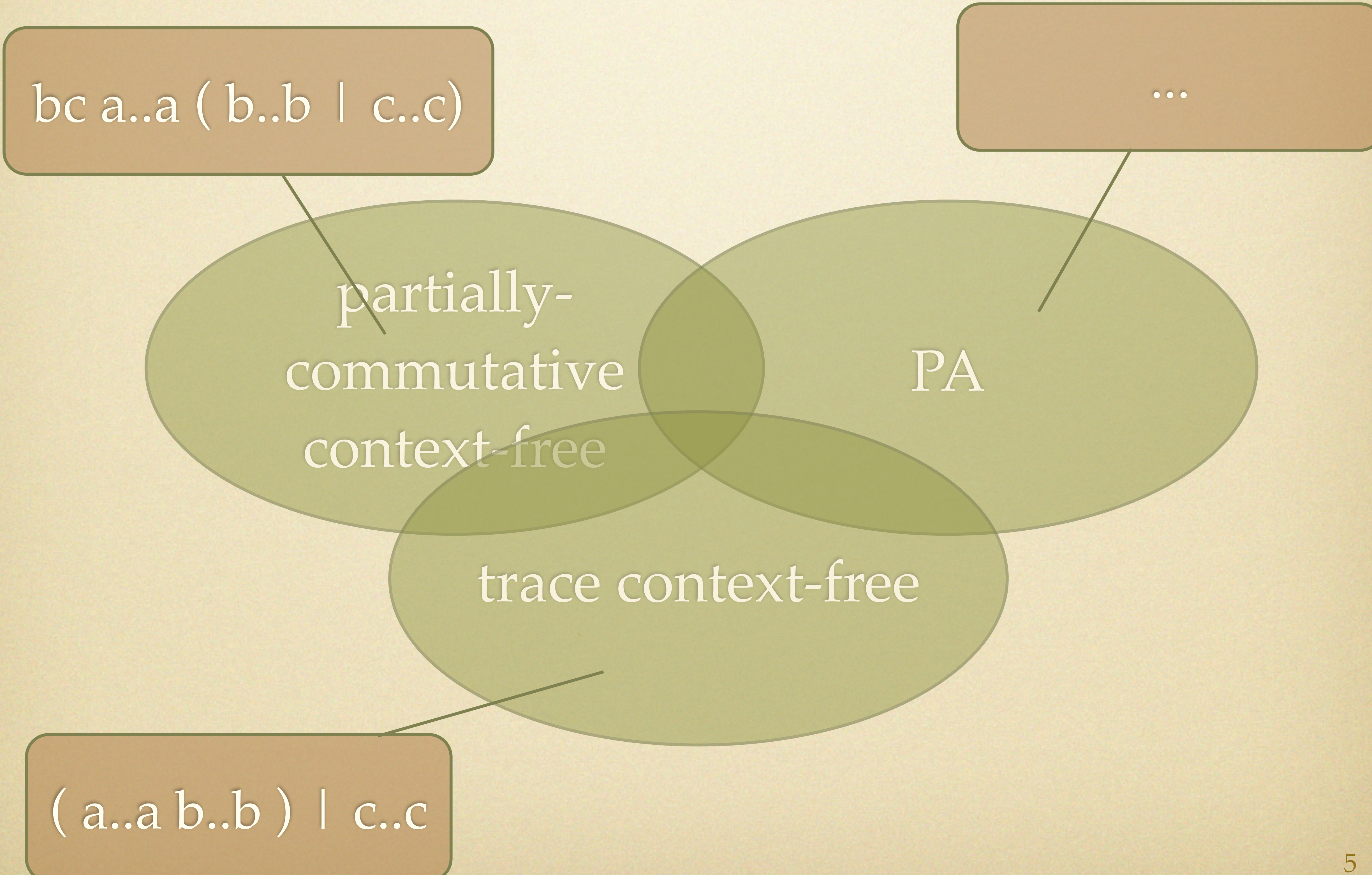
Expressibility



Expressibility



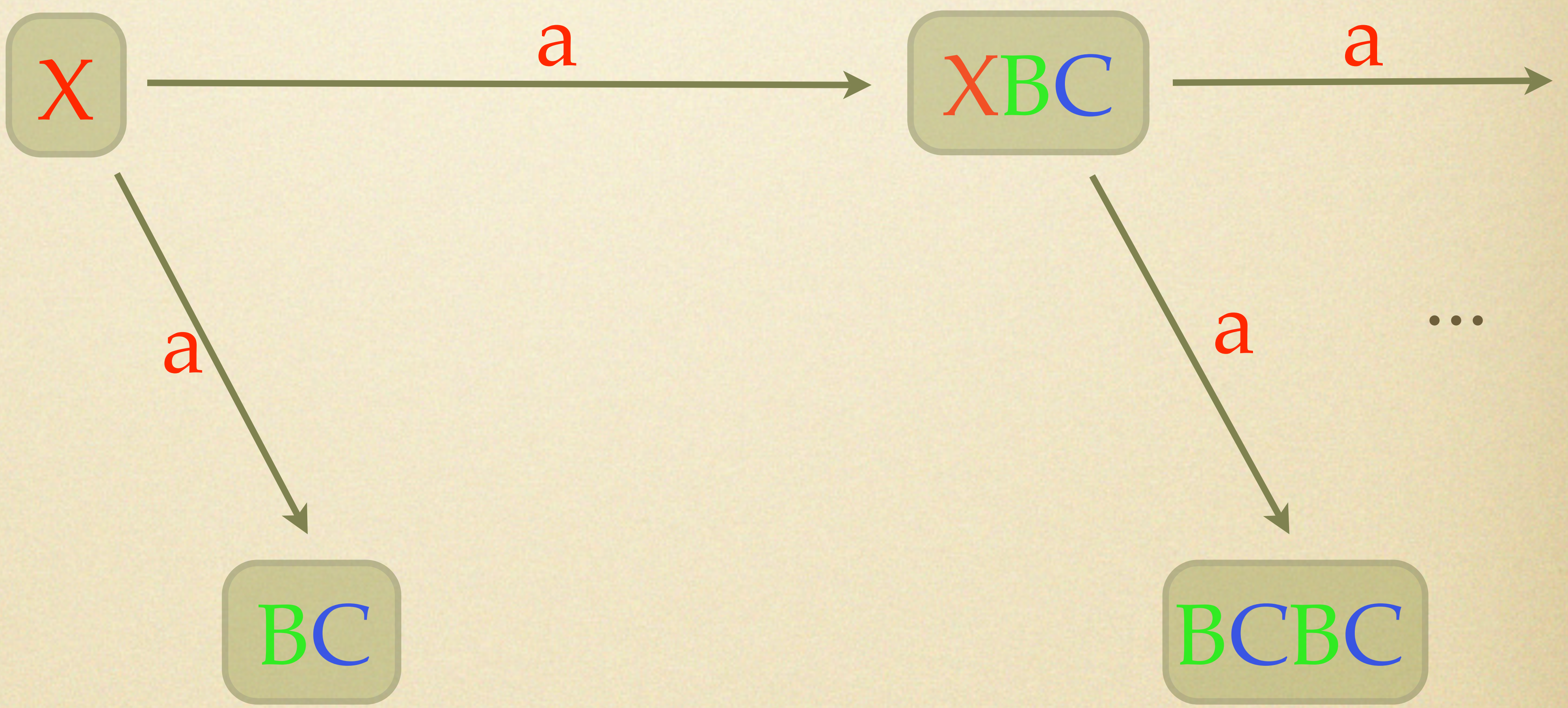
Expressibility



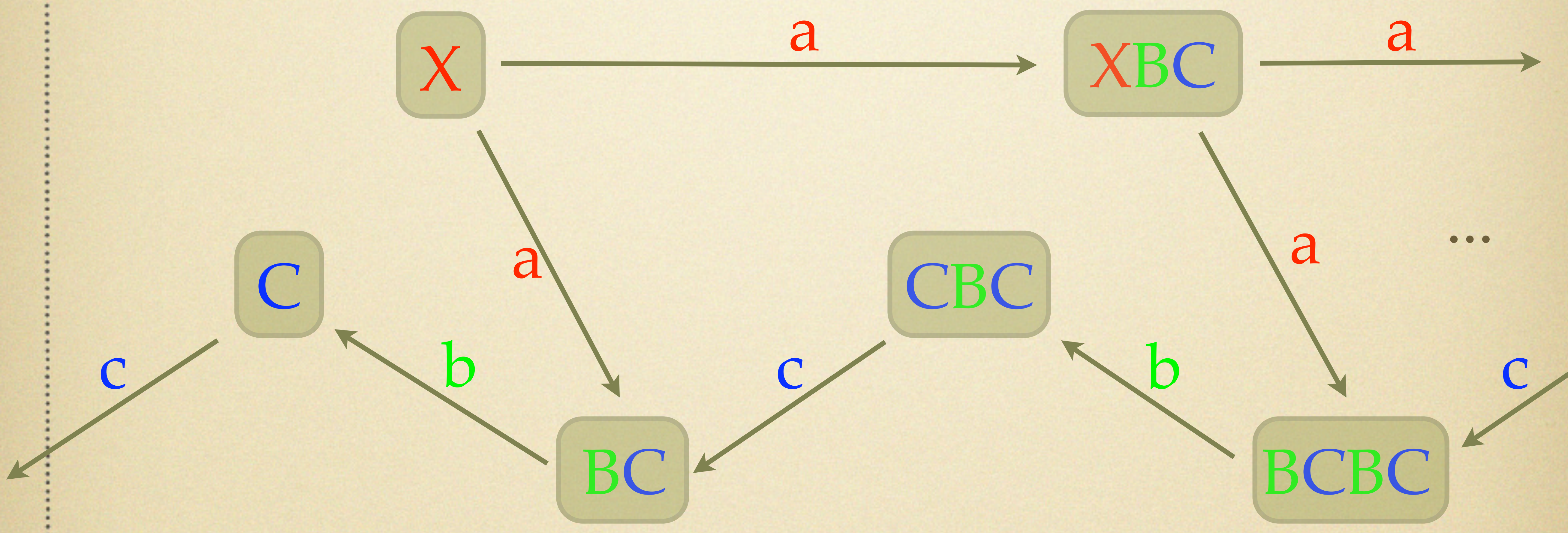
context-free processes

non-terminal = elementary process

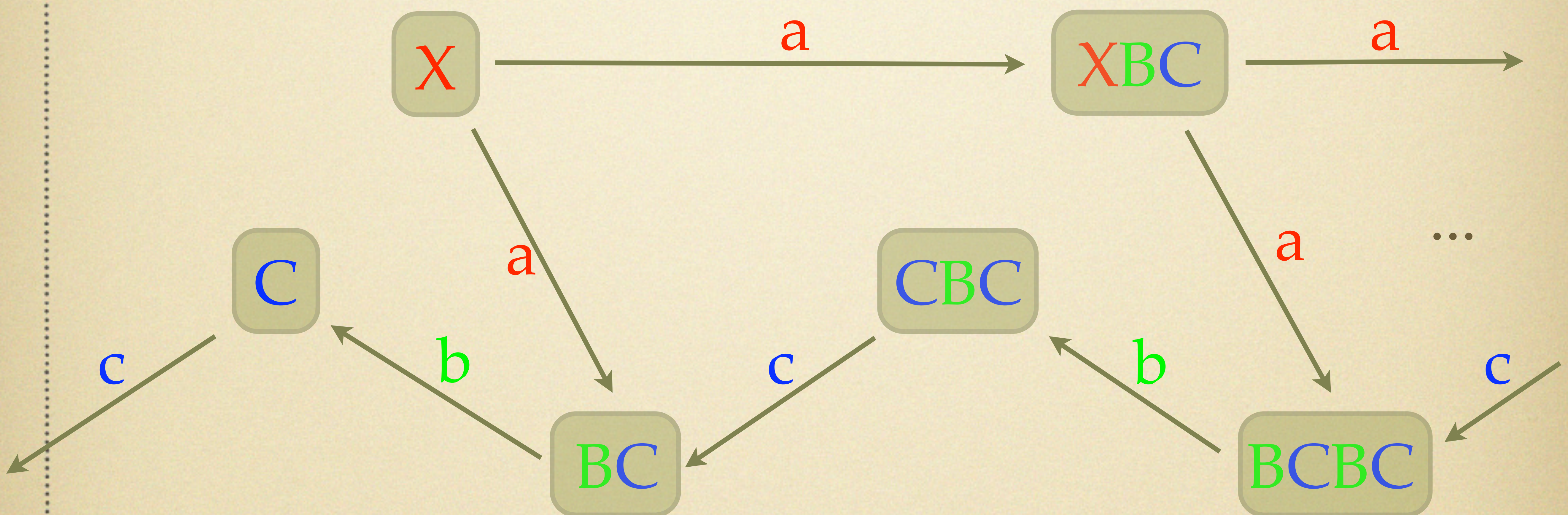
context-free processes



context-free processes

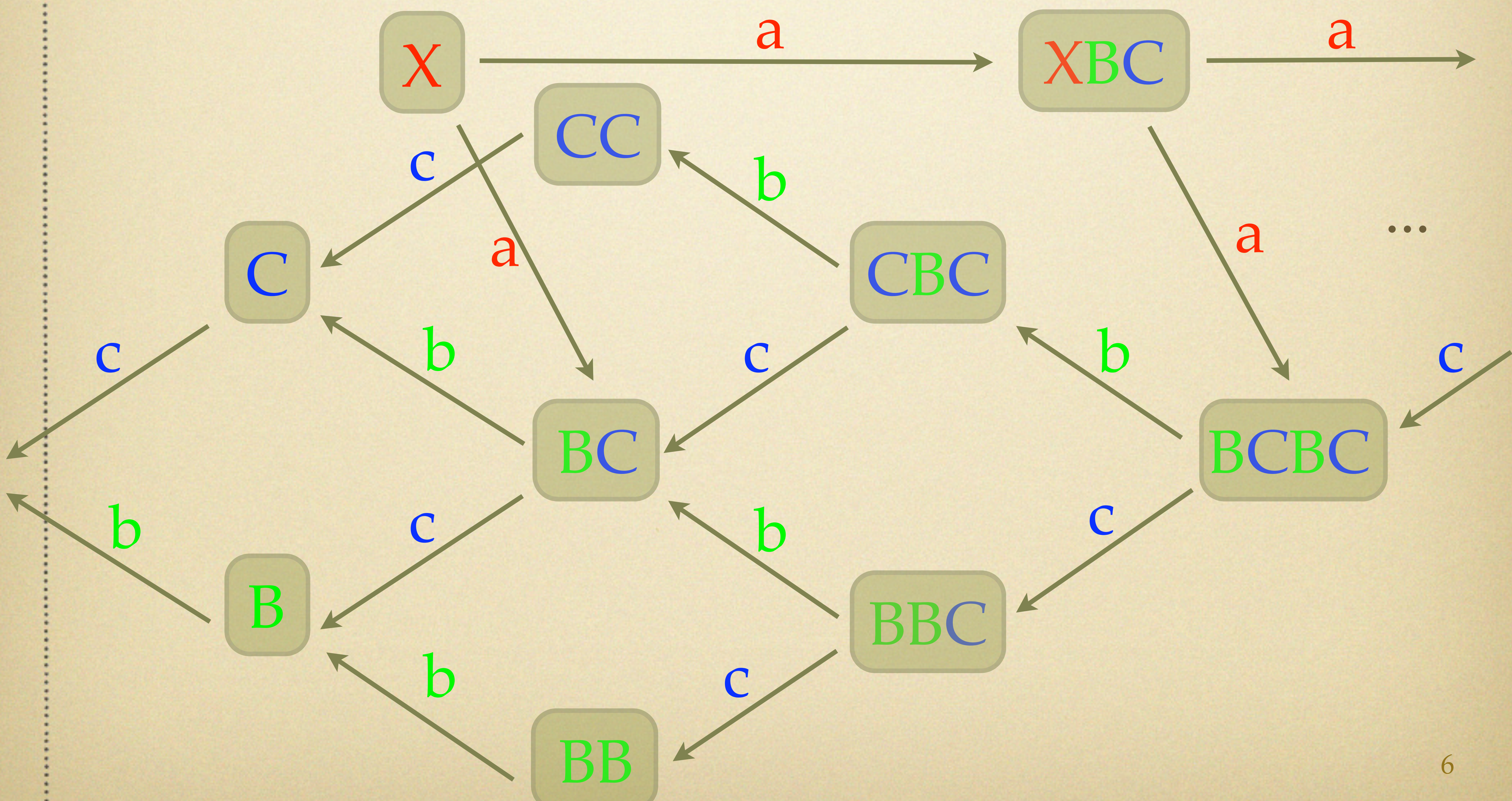


Partially-commutative context-free processes

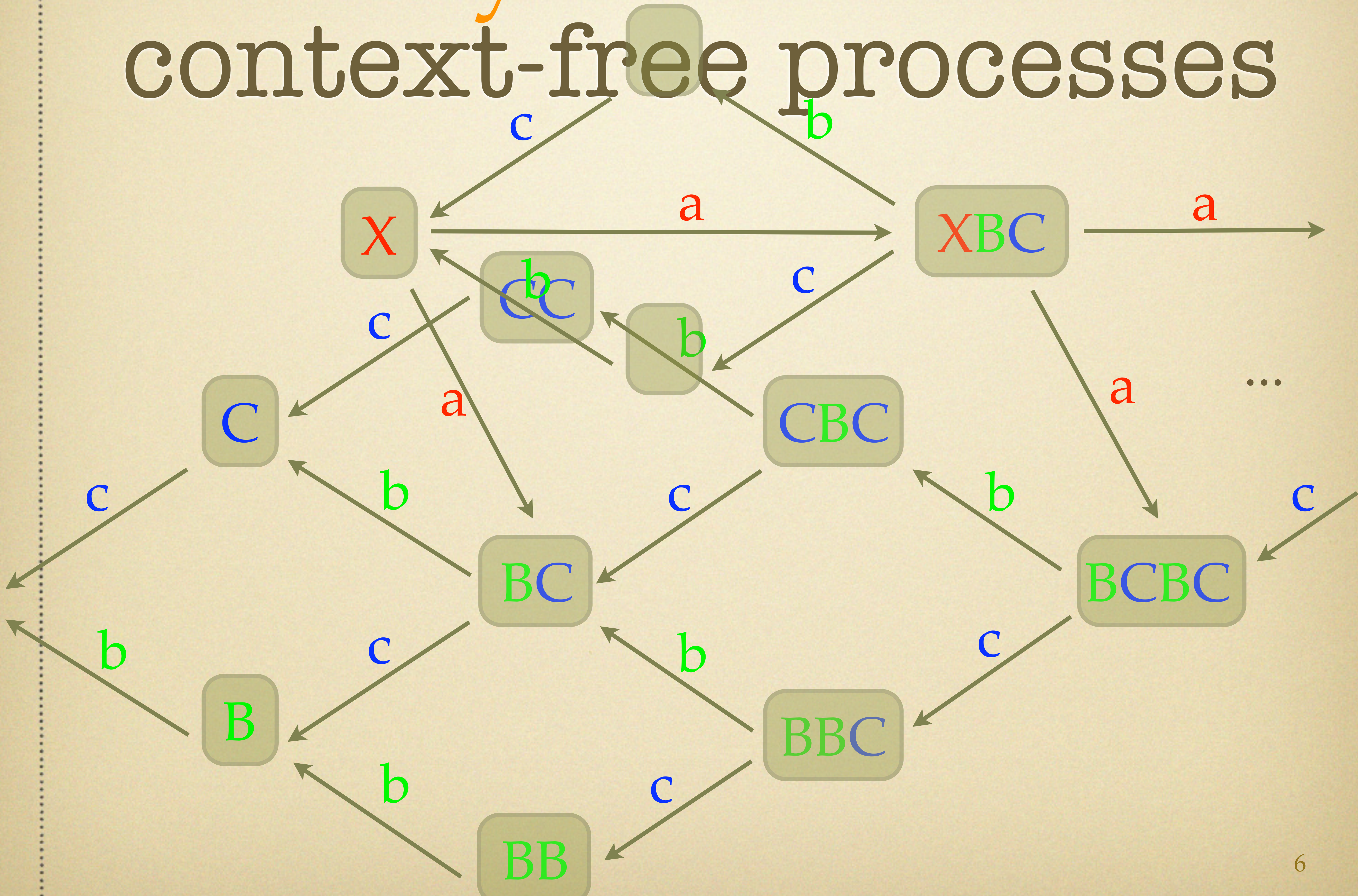


B and C independent

Partially-commutative context-free processes



Partially-commutative context-free processes



Transition rules

$w, v \in V^*$

$$X\,w \xrightarrow{a} v\,w$$

if there is a production $X \xrightarrow{a} v$

Transition rules

up to transposition of independent non-terminals

$w, v \in V^*$

$$X\, w \xrightarrow{a} v\, w$$

if there is a production $X \xrightarrow{a} v$

Transition rules

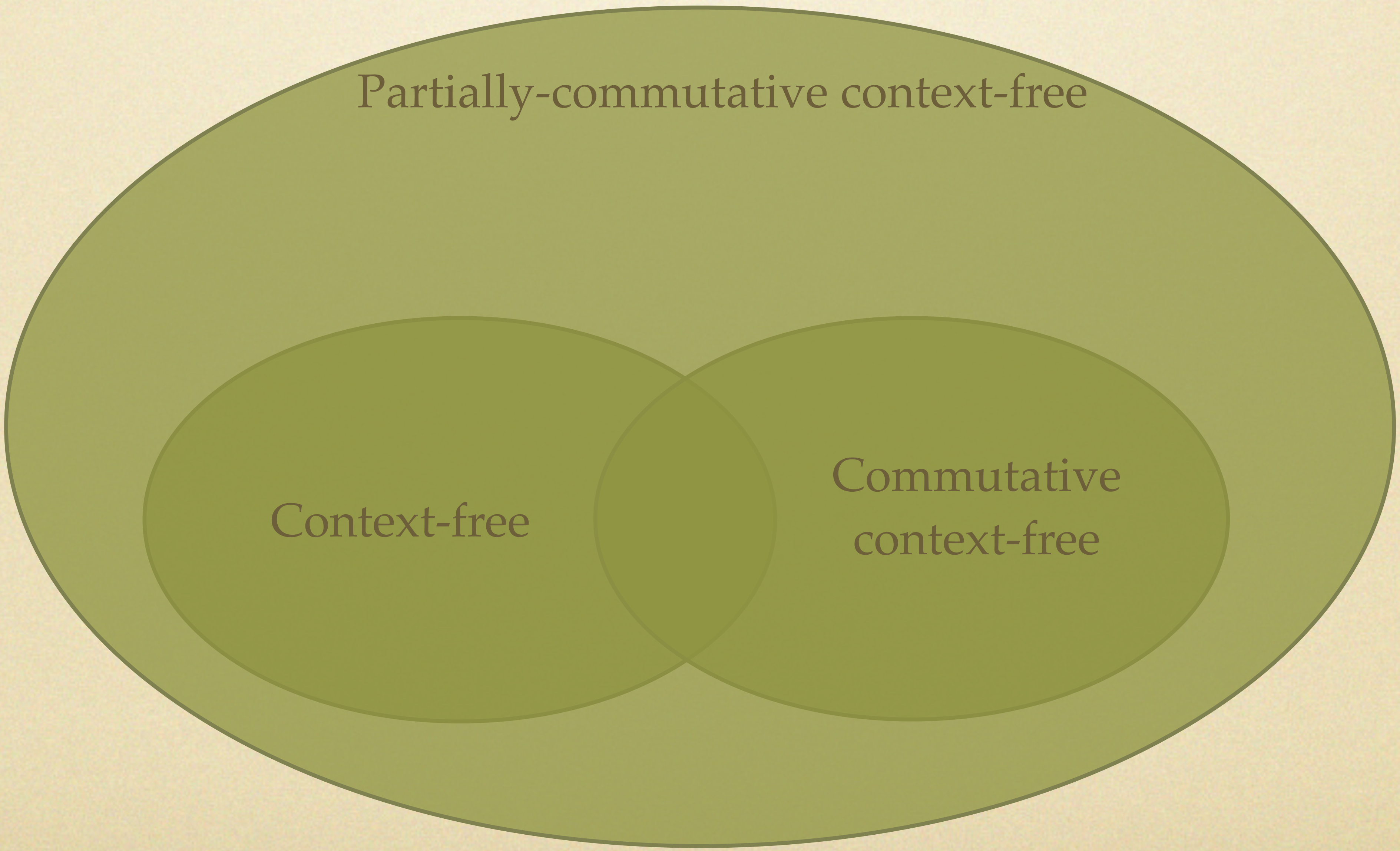
up to transposition of independent non-terminals

$$X\,w \xrightarrow{a} v\,w \qquad w, v \in V^*$$

if there is a production $X \xrightarrow{a} v$

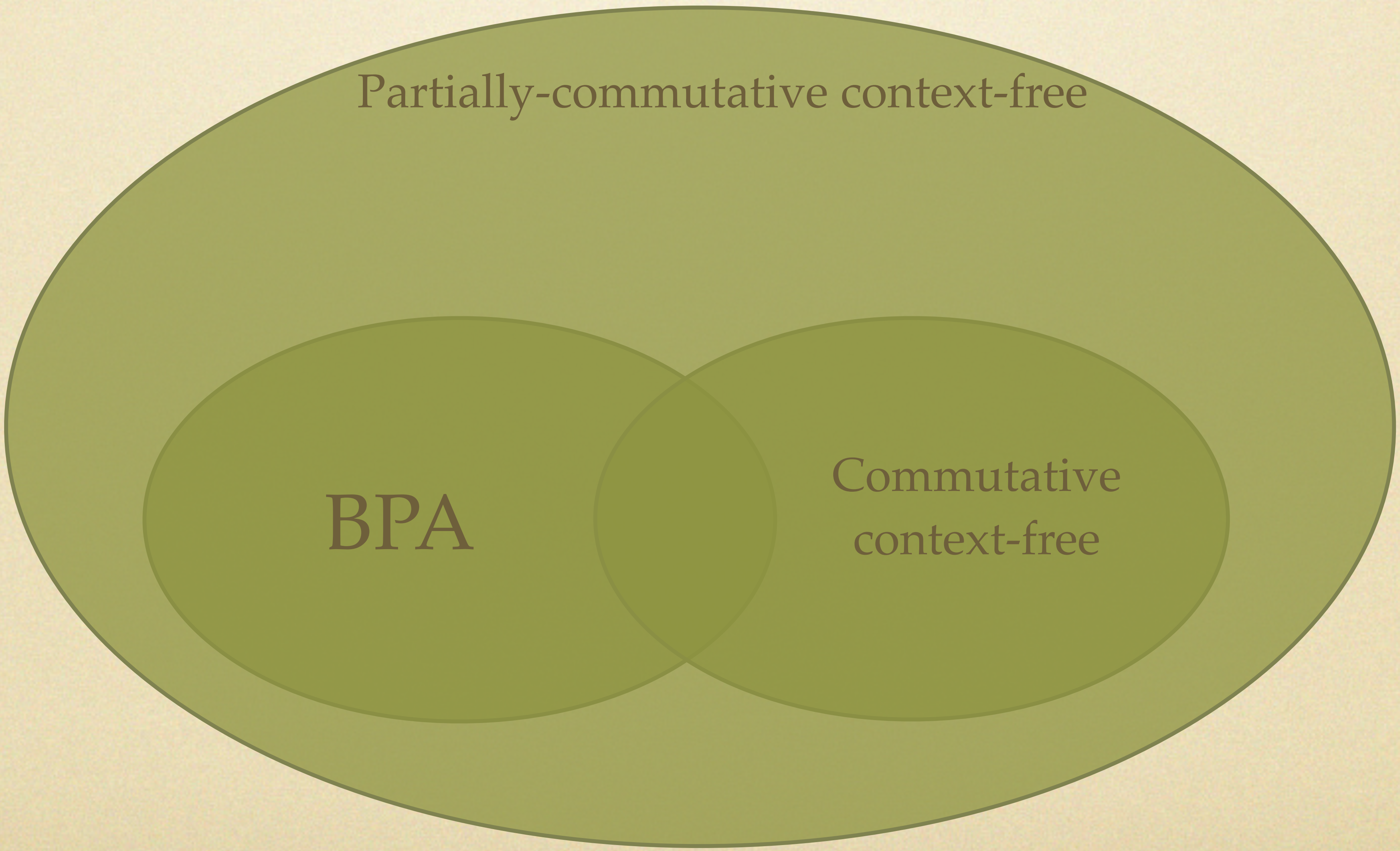
$$\boxed{\text{process} = \text{trace over } (V, I)}$$

BPC



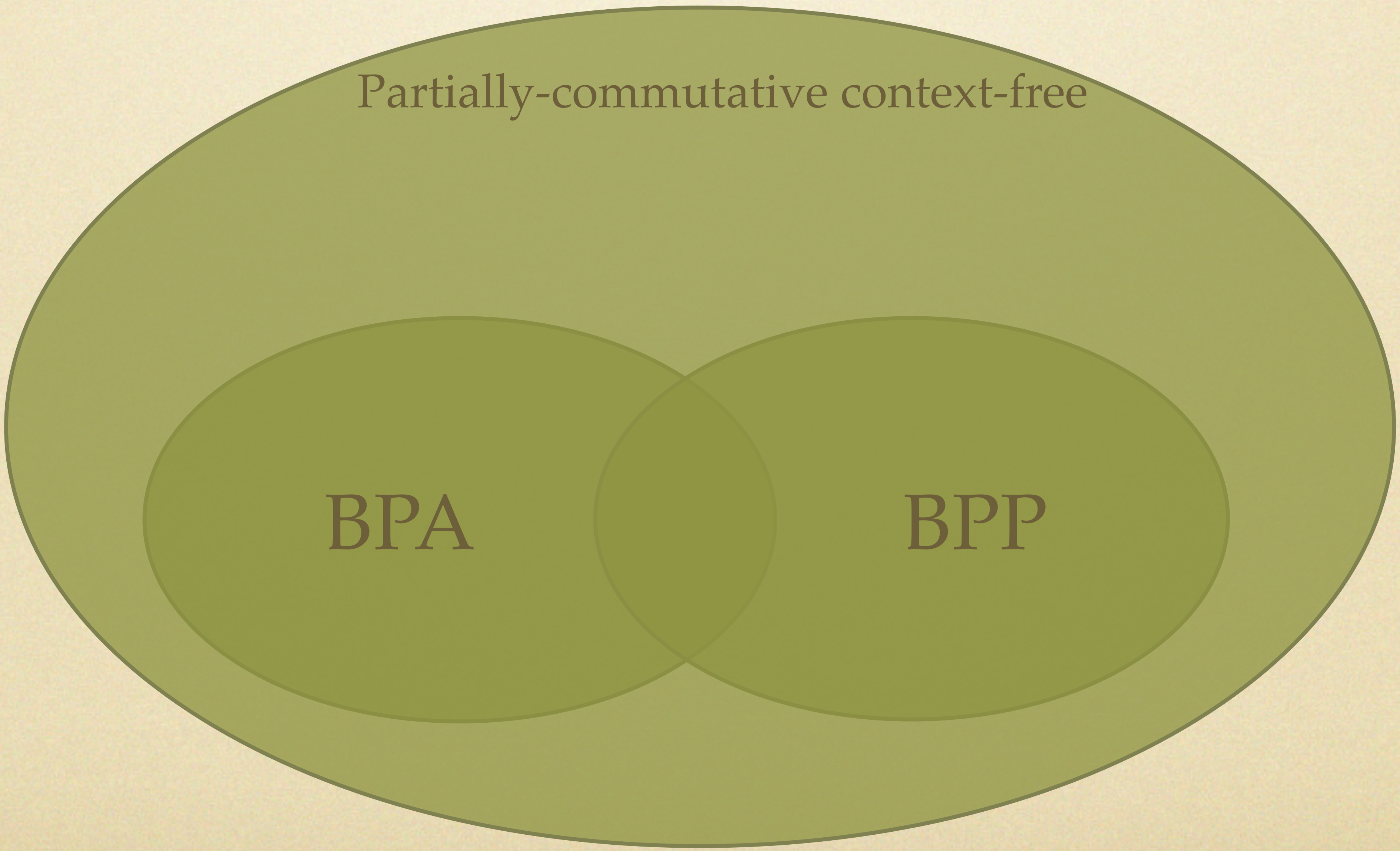
Partially-commutative context-free
Context-free
Commutative
context-free

BPC

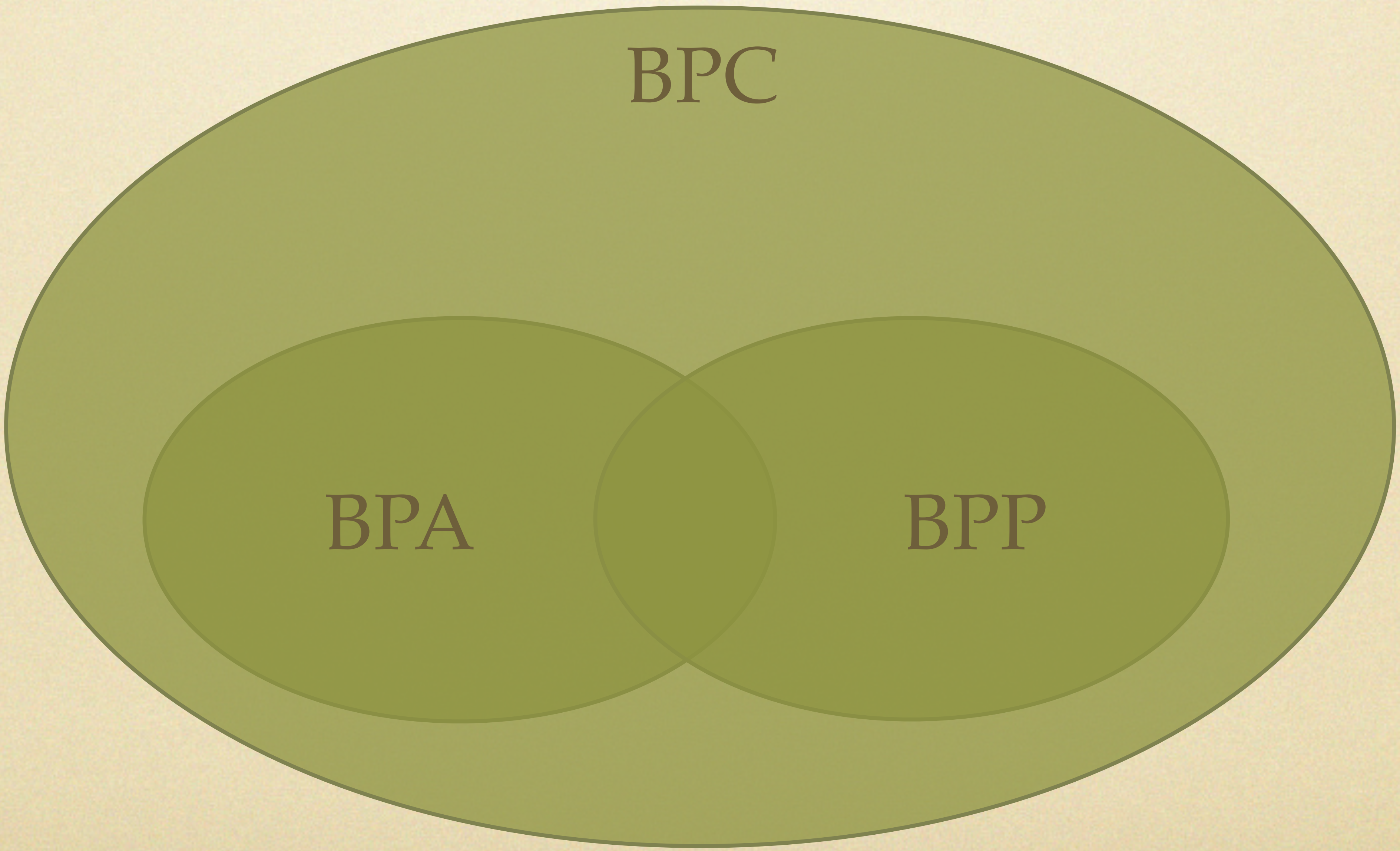


Partially-commutative context-free
BPA
Commutative
context-free

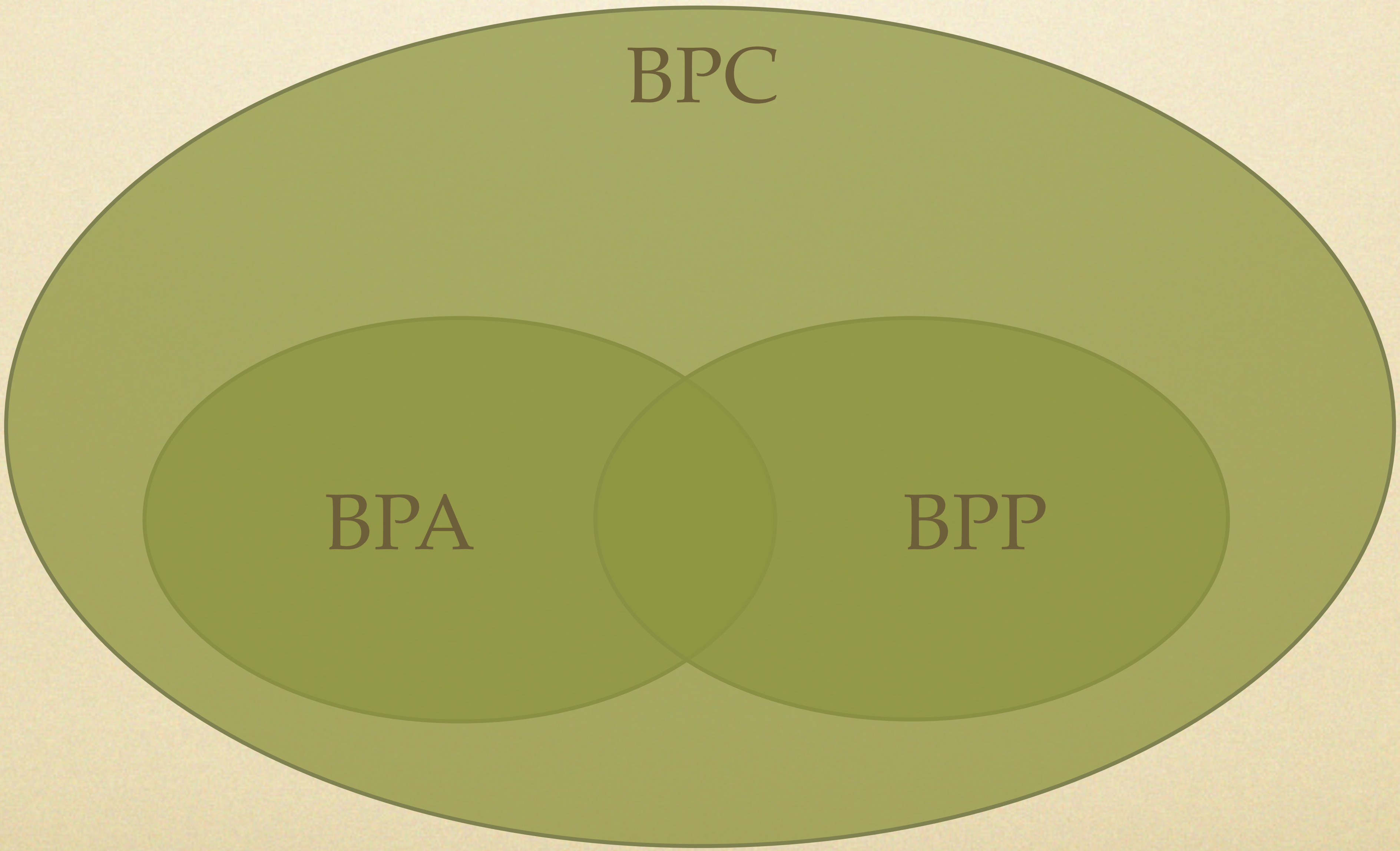
BPC



BPC

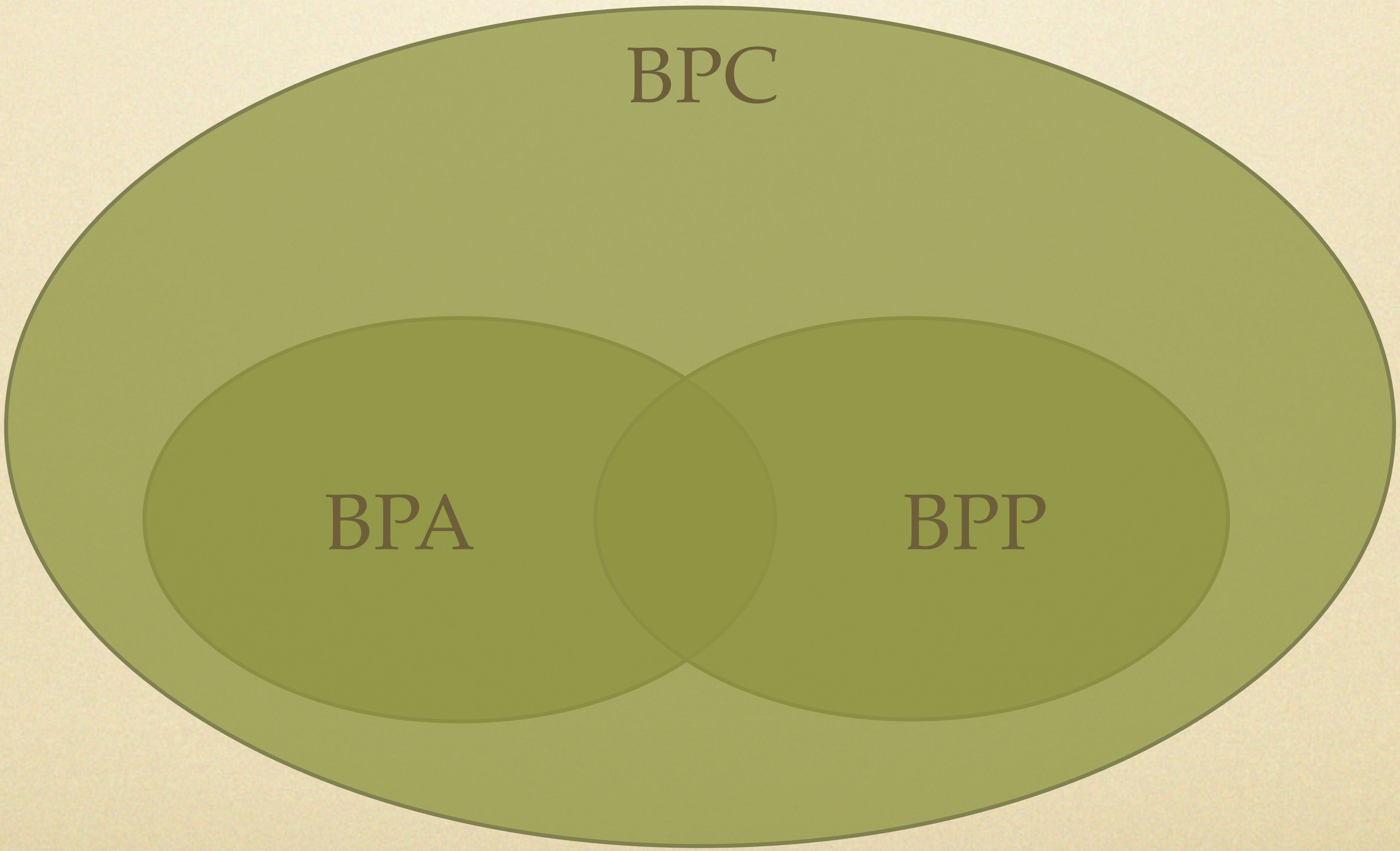


Transitive BPC



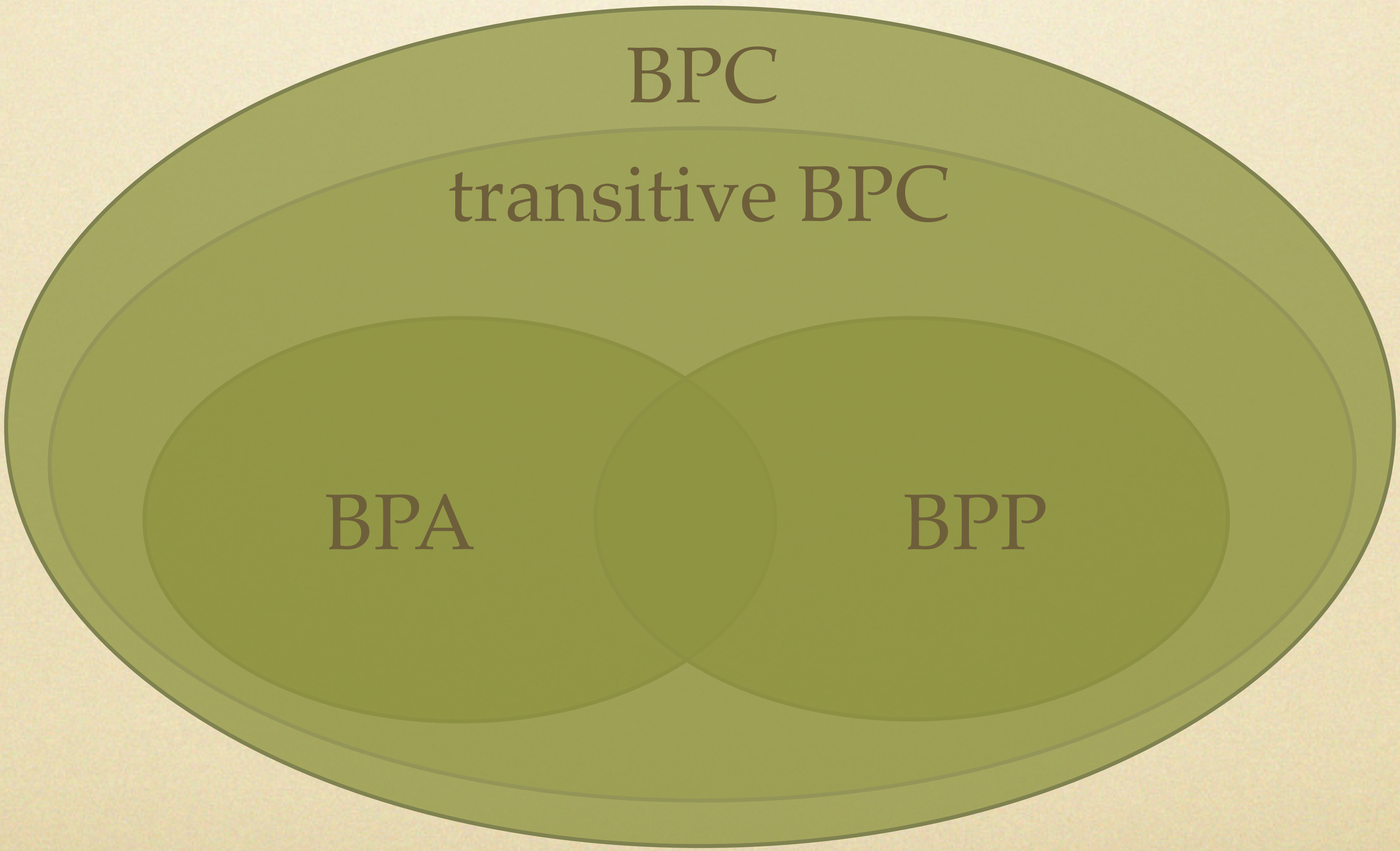
Transitive BPC

transitive dependence $D = V^2 \setminus I$



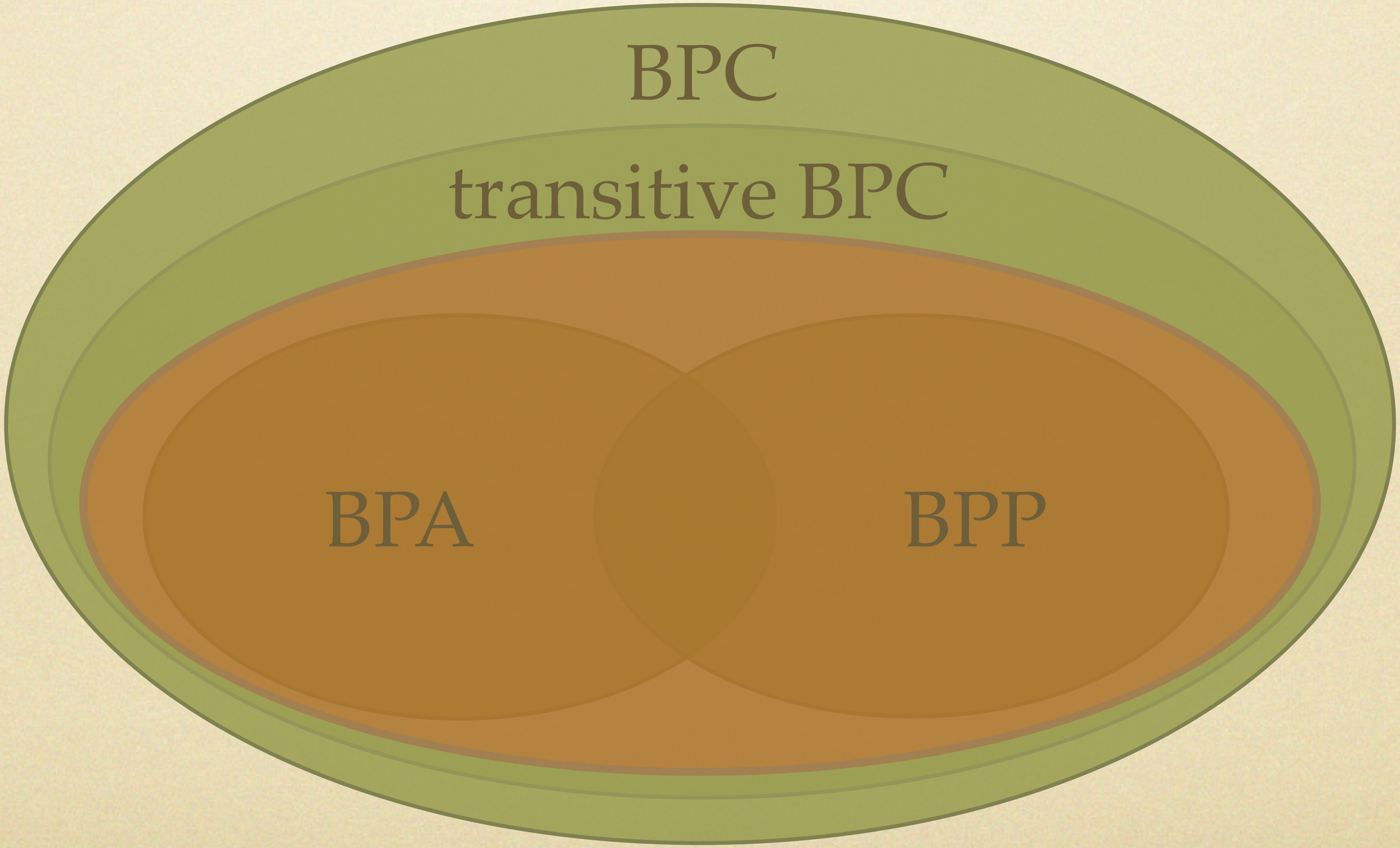
Transitive BPC

transitive dependence $D = V^2 \setminus I$



Transitive BPC

transitive dependence $D = V^2 \setminus I$



Transitive BPC - example

$X \xrightarrow{a} XBC$

$B \xrightarrow{b}$

$X \xrightarrow{a} BC$

$C \xrightarrow{c}$

Transitive BPC - example

$$I = \{ (X, C), (B, C) \}$$

$$X \xrightarrow{a} XBC \qquad B \xrightarrow{b}$$

$$X \xrightarrow{a} BC \qquad C \xrightarrow{c}$$

Transitive BPC - example

$$I = \{ (X, C), (B, C) \}$$

$$D = \{ \{X, B\}, \{C\} \}$$

$$X \xrightarrow{a} XBC \qquad B \xrightarrow{b}$$

$$X \xrightarrow{a} BC \qquad C \xrightarrow{c}$$

Transitive BPC - example

"threads"

$I = \{ (X, C), (B, C) \}$

$D = \{ \{X, B\}, \{C\} \}$

$X \xrightarrow{a} XBC$

$B \xrightarrow{b}$

$X \xrightarrow{a} BC$

$C \xrightarrow{c}$

Transitive BPC - example

“threads”

$I = \{ (X, C), (B, C) \}$

$D = \{ \{X, B\}, \{C\} \}$

$X \xrightarrow{a} XBC \qquad B \xrightarrow{b}$

$X \xrightarrow{a} BC \qquad C \xrightarrow{c}$

language = (a .. a b .. b) | c .. c,

a “preceeds” c

Outline

- What is “partially-commutative context-free” ?
- What is “processes“ ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

Outline

- What is "partially-commutative context-free" ?
- What is "processes" ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

Strong bisimilarity

normed processes

Strong bisimilarity

each non-terminal generates a word

normed processes

Strong bisimilarity

each elementary process may terminate

normed processes

Strong bisimilarity

each elementary process may terminate

normed processes

- on normed BPA and BPP, bisimulation is in P [Hirshfeld, Jerrum, Moller '96]

Strong bisimilarity

each elementary process may terminate

normed processes

- on normed BPA and BPP, bisimulation is in P [Hirshfeld, Jerrum, Moller '96]
- on normed PA, bisimulation is in 2-NEXPTIME [Hirshfeld, Jerrum '99]

Strong bisimilarity

each elementary process may terminate

normed processes

- on normed BPA and BPP, bisimulation is in P [Hirshfeld, Jerrum, Moller ‘96]
- on normed PA, bisimulation is in 2-NEXPTIME [Hirshfeld, Jerrum ’99]
- BPA ~ BPP is in P [Jančar, Kot, Sawa ‘08]

Strong bisimilarity

Challenge 1:

to extend the tractable class

Challenge 2:

BPA and BPP algorithms are totally different

Contribution

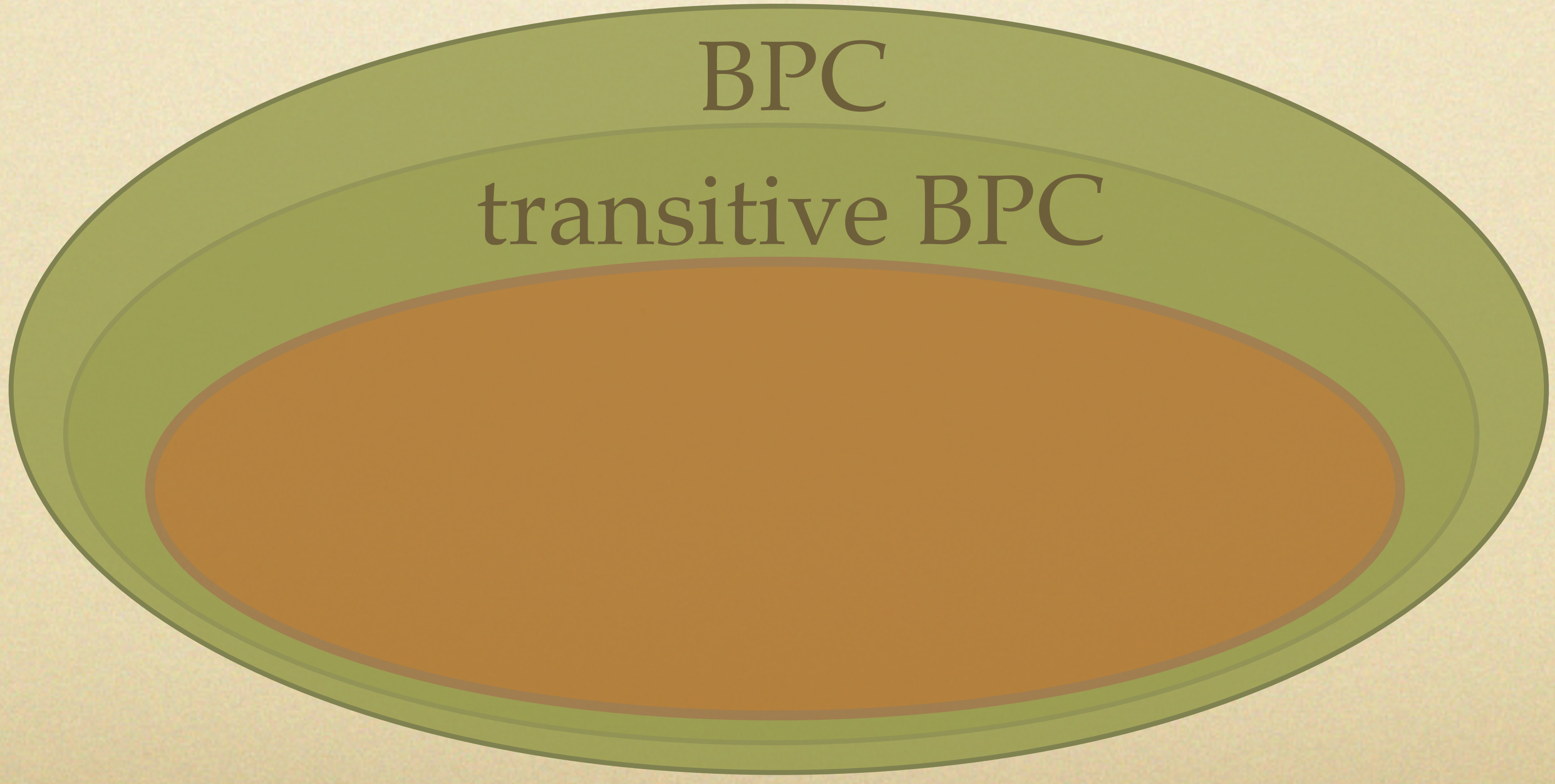
Theorem:

Bisimilarity is decidable in polynomial time in a subclass of transitive BPC

Contribution

Theorem:

Bisimilarity is decidable in polynomial time in a subclass of transitive BPC



Contribution

Theorem:

Bisimilarity is decidable in polynomial time in a subclass of transitive BPC

Remark:

One polynomial-time algorithm for both BPA and BPP

Contribution

Idea:

The BPP algorithm works for BPA just as well !

Contribution

Idea:

The BPP algorithm works for BPA just as well !

Naive implementation in exponential time

Contribution

Idea:

The BPP algorithm works for BPA just as well !

Naive implementation in exponential time

Compression of strings helps

Contribution

Idea:

The BPP algorithm works for BPA just as well !

Naive implementation is exponential time

Compression of strings helps

Outline

- What is “partially-commutative context-free” ?
- What is “processes“ ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

Outline

- What is “partially-commutative context-free” ?
- What is “processes“ ?
- Strong bisimilarity checking
- Our contribution
- Outline of the algorithm
- Further research

Outline of the algorithm

Bisimilarity ~ is a congruence with

the unique decomposition property

Outline of the algorithm

Bisimilarity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Outline of the algorithm

Bisimilarity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Algorithm computes iteratively the bisimilarity base

Outline of the algorithm

in BPC it isn't

Bisimilarity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Algorithm computes iteratively the bisimilarity base

Outline of the algorithm

Bisimilarity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Algorithm computes iteratively the bisimilarity base

Unique decomposition

like prime decomposition of natural numbers

Unique decomposition

Each non-terminal X is either:

Unique decomposition

Each non-terminal X is either:

$\alpha \neq X$

- decomposable: $X \sim \alpha$, or

Unique decomposition

Each non-terminal X is either:

- decomposable: $X \sim \alpha$, or
- non-decomposable, or prime

Unique decomposition

Each non-terminal X is either:

- decomposable: $X \sim \alpha$, or
- non-decomposable, or prime

Each process decomposes uniquely into primes

Unique decomposition

Each non-terminal X is either:

- decomposable: $X \sim \alpha$, or
- non-decomposable, or prime

Each process decomposes uniquely into primes

Makes sense for congruences other than $\sim$

Cancellation

if $\alpha\, \gamma \sim \beta\, \gamma$ then $\alpha \sim \beta$

Cancellation

if $\alpha\,\gamma \sim \beta\,\gamma$ then $\alpha \sim \beta$

follows from the unique decomposition

Decomposition vs cancellation

BPA

BPP

Decomposition vs cancellation

BPA

cancellation

BPP

Decomposition vs cancellation

BPA

cancellation

$\downarrow$

decomposition

BPP

Decomposition vs cancellation

BPA

cancellation

decomposition

BPP

decomposition

Decomposition vs cancellation

BPA

cancellation

$\downarrow$

decomposition

BPP

decomposition

$\downarrow$

cancellation

Decomposition vs cancellation

BPA

cancellation

↓

decomposition

BPC ?

BPP

decomposition

↓

cancellation

Decomposition vs cancellation

BPC weak cancellation

Decomposition vs cancellation

BPC weak cancellation

decomposition

Decomposition vs cancellation

BPC

weak cancellation

decomposition

cancellation

Decomposition vs cancellation

Theorem:

Each congruence on transitive BPC that is:

Decomposition vs cancellation

Theorem:

Each congruence on transitive BPC that is:

- norm-reducing bisimulation

Decomposition vs cancellation

Theorem:

Each congruence on transitive BPC that is:

- norm-reducing bisimulation
- weakly cancellative

Decomposition vs cancellation

Theorem:

Each congruence on transitive BPC that is:

- norm-reducing bisimulation
- weakly cancellative

has unique decomposition property

Decomposition vs cancellation

not only bisimilarity

Theorem:

Each congruence on transitive BPC that is:

- norm-reducing bisimulation
- weakly cancellative

has unique decomposition property

Outline of the algorithm

Bisimularity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Algorithm computes iteratively the bisimilarity base

Outline of the algorithm

Bisimularity ~ is a congruence with

the unique decomposition property

Hence it is representable by a finite base

Algorithm computes iteratively the bisimilarity base

Base

Base B:

Base

a succinct representation of a congruence with unique decomposition property

Base B:

Base

a succinct representation of a congruence
with unique decomposition property

Base B:

- primes $\subseteq$ V

Base

a succinct representation of a congruence with unique decomposition property

Base B:

- primes $\subseteq$ V
- decompositions $X = \alpha$ into primes, one for each non-prime X

Bisimulation approximants

BPA

BPP

Bisimulation approximants

BPA

initialization:

$B_0 \supseteq B_{\sim}$

BPP

Bisimulation approximants

BPA

initialization:

$B_0 \supseteq B_\sim$

BPP

initialization:

B_0 represents norm-equality

Bisimulation approximants

BPA

initialization:

$B_0 \supseteq B_{\sim}$

refinement:

removing pairs from B

BPP

initialization:

B_0 represents norm-equality

Bisimulation approximants

BPA

initialization:

$B_0 \supseteq B_{\sim}$

refinement:

removing pairs from B

BPP

initialization:

B_0 represents norm-equality

refinement:

bisimulation "expansion"

Bisimulation approximants

BPA

BPP

BPC ?

initialization:

$B_0 \supseteq B_\sim$

initialization:

B_0 represents norm-equality

refinement:

removing pairs from B

refinement:

bisimulation "expansion"

Bisimulation approximants

BPA

initialization:

$B_0 \supseteq B_\sim$

refinement:

removing pairs from B

BPC ?

BPP

initialization:

B_0 represents norm-equality

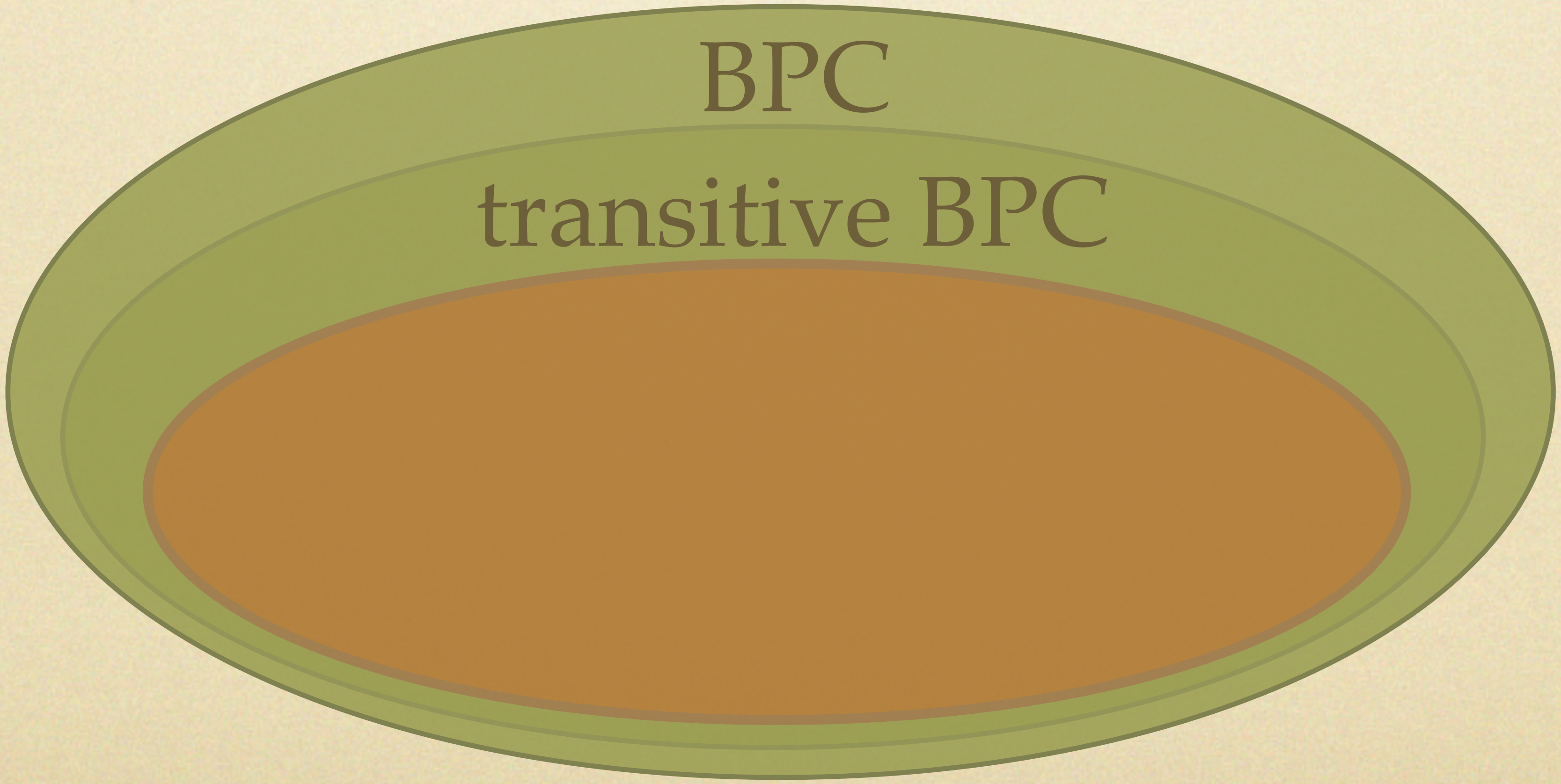
refinement:

bisimulation "expansion"

Further research

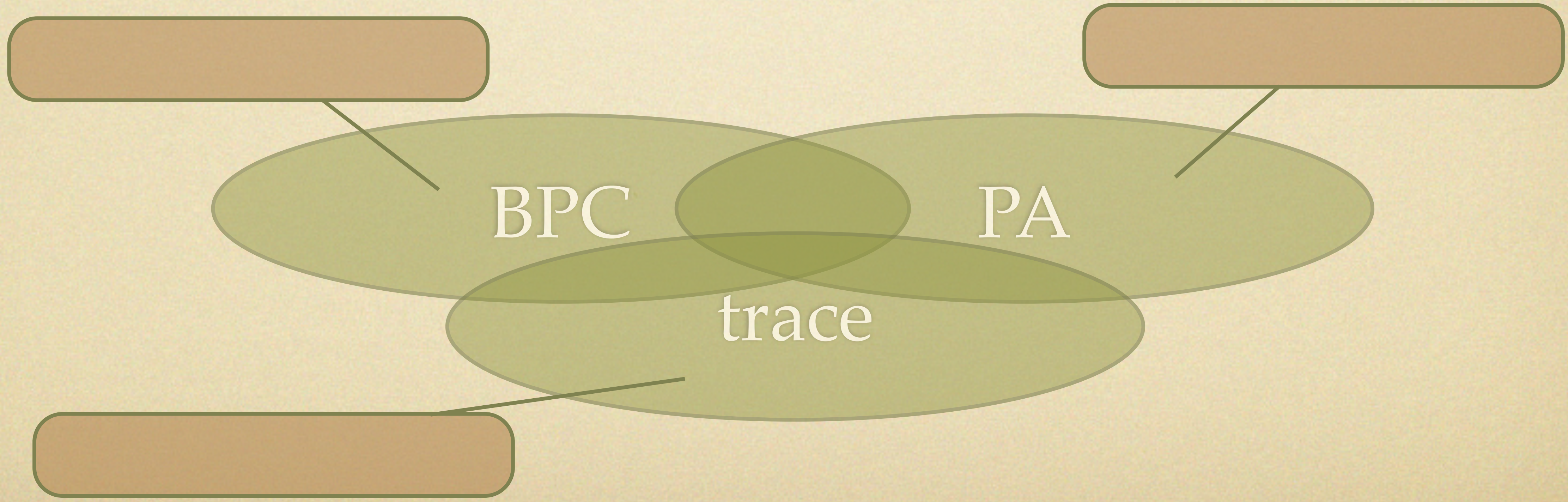
Further research

- beyond transitive BPC



Further research

- beyond transitive BPC
- expressibility of "partially-commutative context-free"



Further research

- beyond transitive BPC
- expressibility of “partially-commutative context-free”
- decidability for non-normed processes

Further research

- beyond transitive BPC
- expressibility of "partially-commutative context-free"
- decidability for non-normed processes
- beyond strong bisimilarity

The algorithm

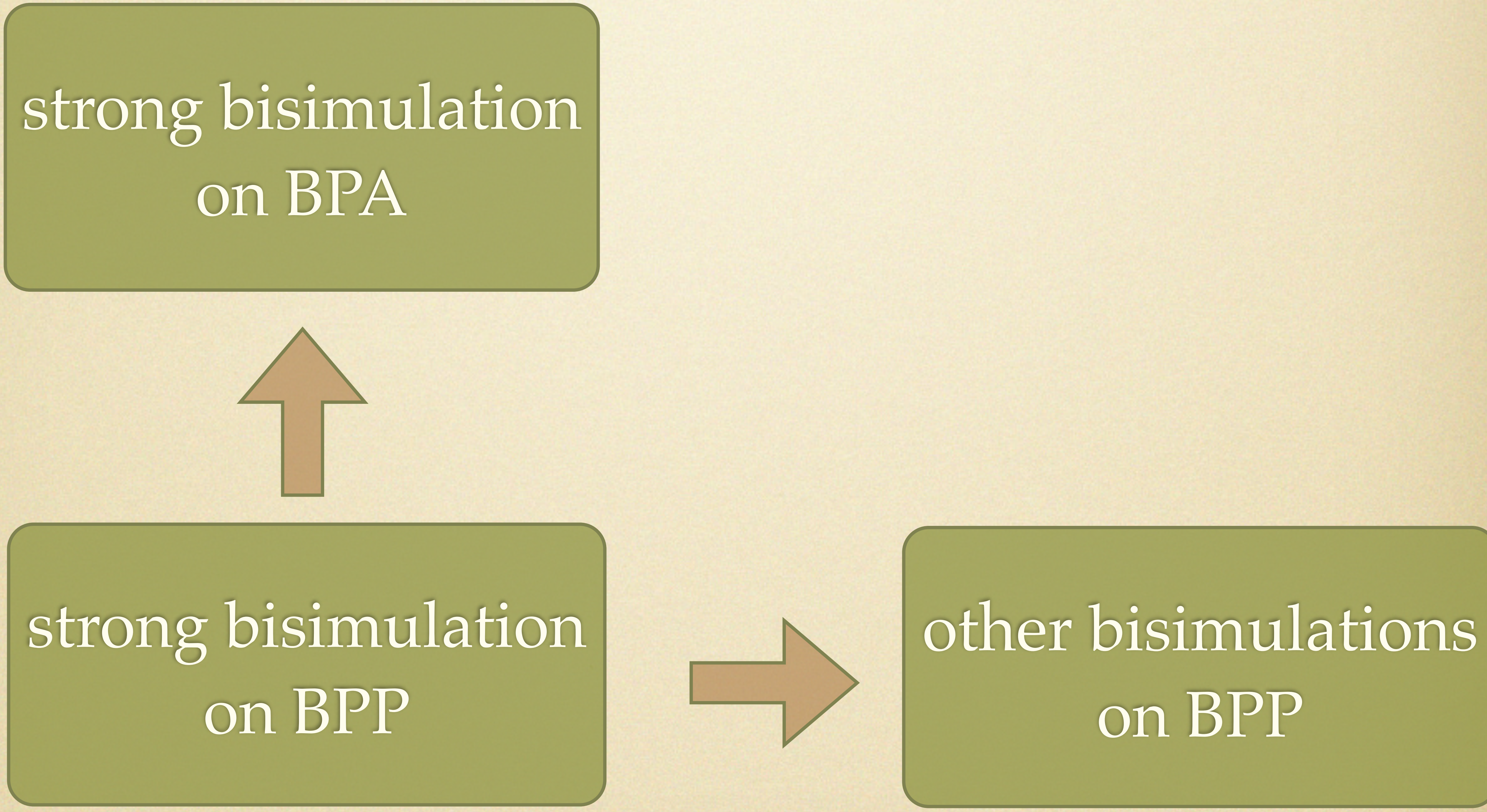
strong bisimulation
on BPP

[Hirshfeld, Jerrum, Moller '96]

The algorithm

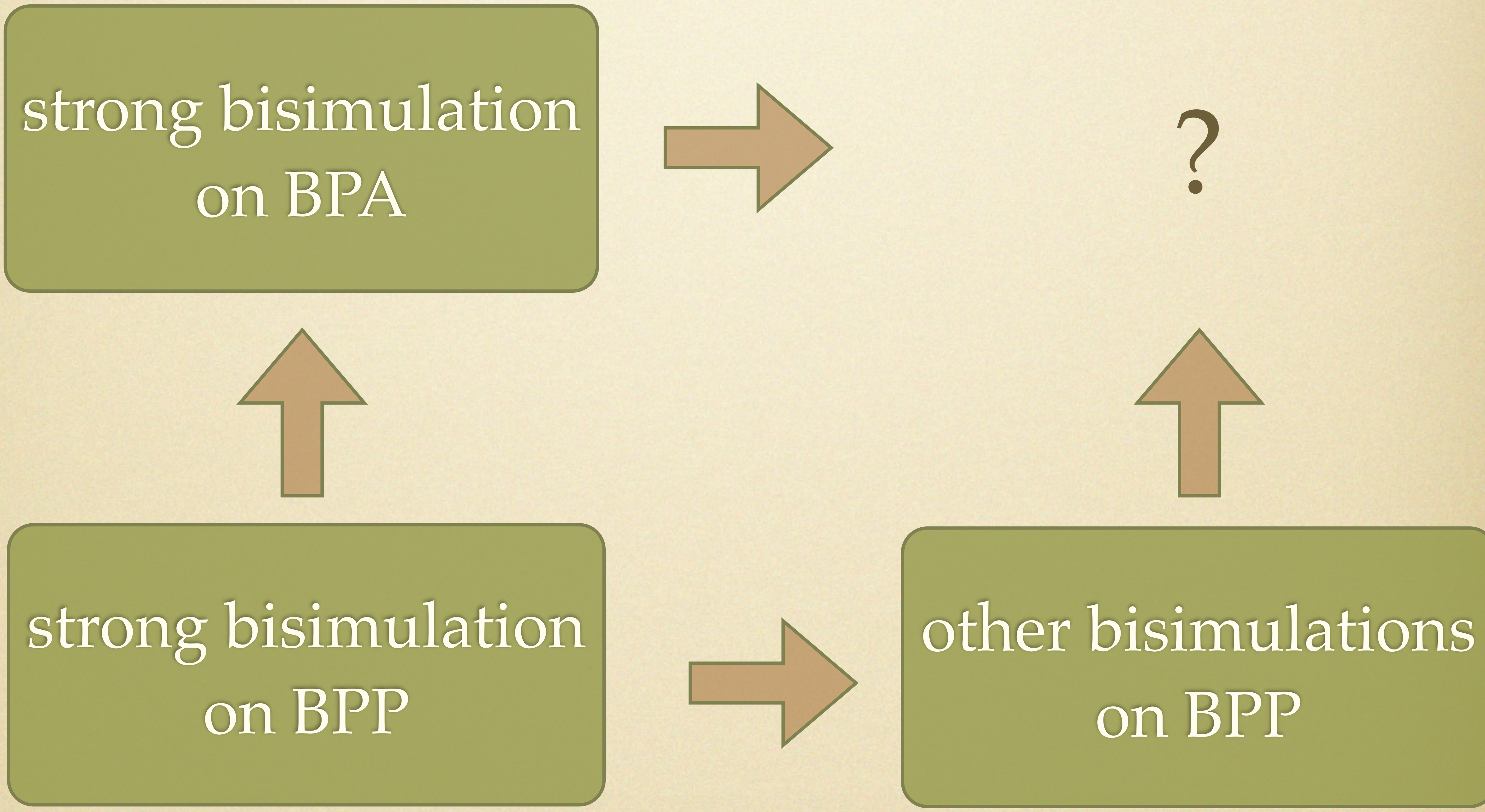
[Hirshfeld, Jerrum, Moller '96]

The algorithm



[Hirshfeld, Jerrum, Moller ’96] [Froeschle, Lasota ’06]

The algorithm



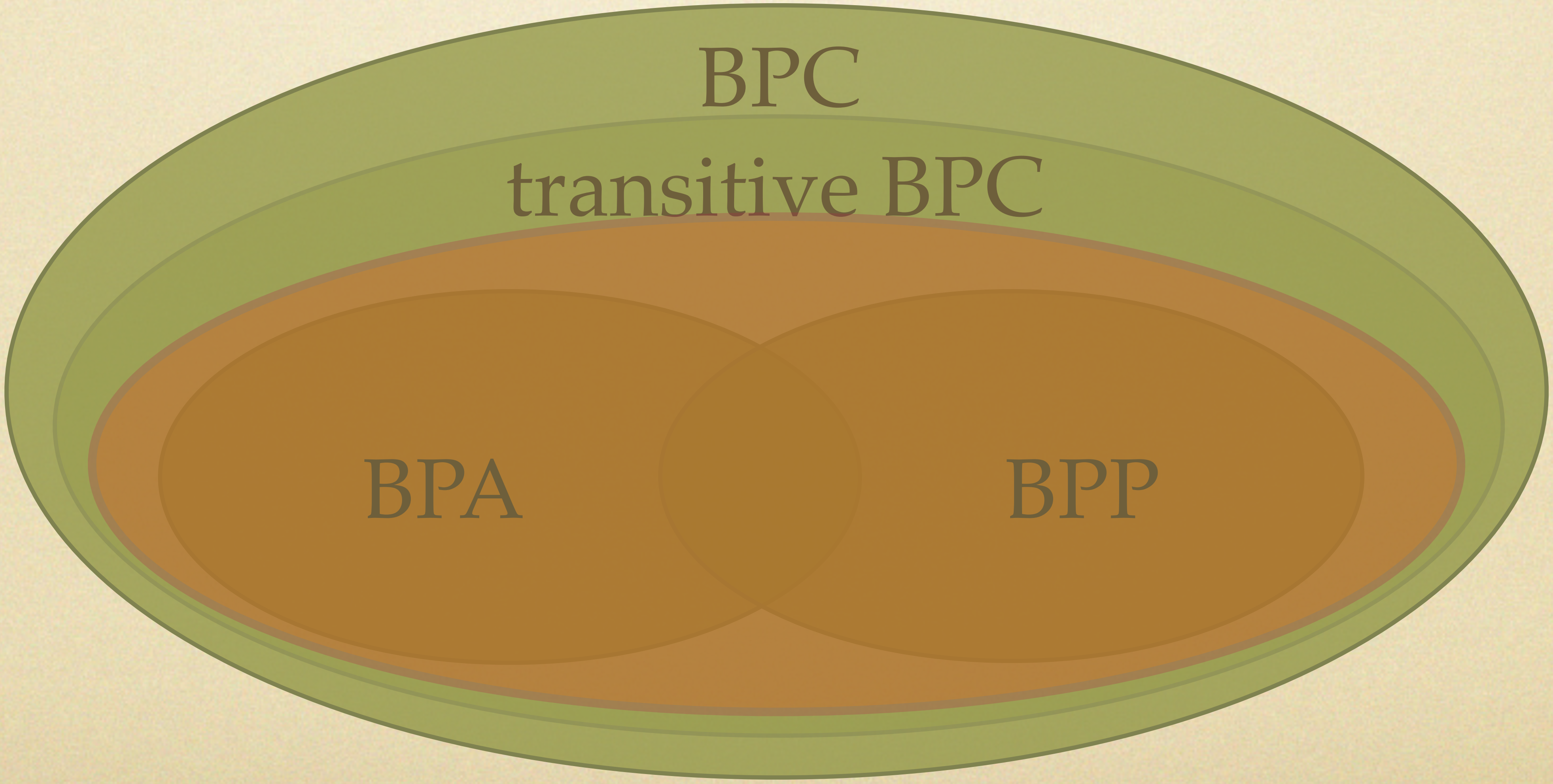
[Hirshfeld, Jerrum, Moller '96]

[Froeschle, Lasota '06]

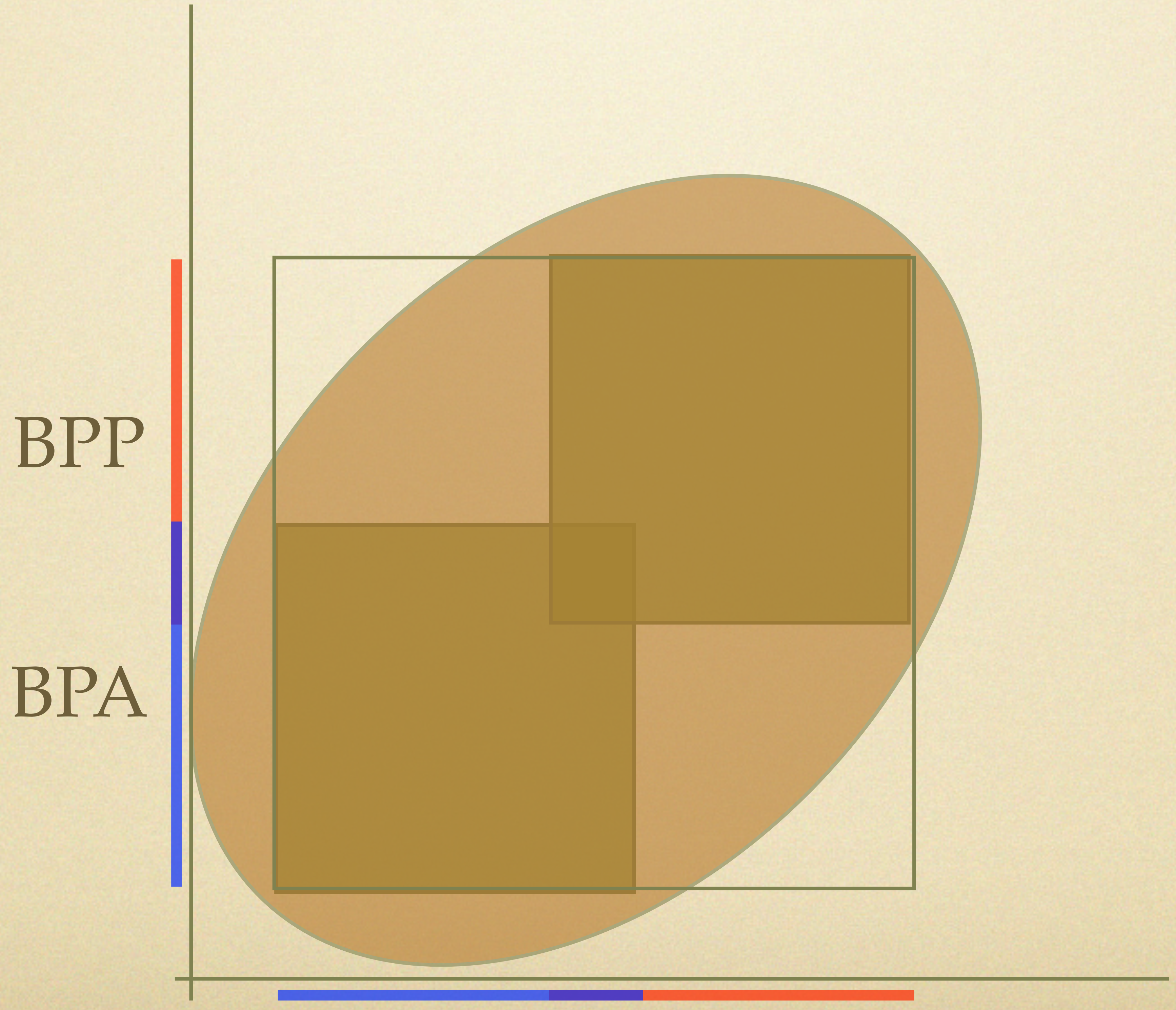
Thanks!

Question:

is BPA ~ BPP subsumed?



The answer: no!



Thanks!