Decomposing Finite Closure Operators by Attribute Exploration

Daniel Borchmann

TU Dresden, Institut für Algebra
Motivation

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context \((G, M, I)\):

- **NextClosure**

\[ \text{input: } M, X \mapsto \rightarrow X \]

- **Close-by-One**

\[ \text{input: } (G, M, I) \]

Can Close-by-One be applied to an arbitrary closure operator \(c\)?
Motivation

Implications and formal contexts
Motivation

Implicitations and formal contexts

formal context $\rightsquigarrow$ few implications

Algorithms to compute the intents of a formal context $(G, M, I)$

NextClosure

input: $M, X \mapsto X''$

Close-by-One

input: $(G, M, I)$

Can Close-by-One be applied to an arbitrary closure operator $c$?
Motivation

Implications and formal contexts

formal context $\rightsquigarrow$ few implications via NextClosure

Algorithms to compute the intents of a formal context $(G, M, I)$

NextClosure input: $M$,

Close-by-One input: $(G, M, I)$

Can Close-by-One be applied to an arbitrary closure operator $c$?
Motivation

Implications and formal contexts

formal context $\rightsquigarrow$ few implications $\rightsquigarrow$ via NextClosure
implications $\rightsquigarrow$ small formal context
Motivation

Implications and formal contexts

formal context $\leadsto$ few implications via NextClosure
implications $\leadsto$ small formal context ?
Motivation

Implications and formal contexts

- formal context $\leadsto$ few implications via NextClosure
- implications $\leadsto$ small formal context ?

How can be compute for a given set of implications a small corresponding formal context?
Motivation

Implications and formal contexts

formal context $\mapsto$ few implications via NextClosure
implications $\mapsto$ small formal context ?

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context $(G, M, I)$
Implications and formal contexts

formal context $\rightsquigarrow$ few implications via NextClosure
implications $\rightsquigarrow$ small formal context

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context $(G, M, I)$

NextClosure
Motivation

Implications and formal contexts

formal context ⇨ few implications via NextClosure
implications ⇨ small formal context  

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context \((G, M, I)\)

\[
\text{NextClosure} \quad \text{input: } M, X \mapsto X''
\]
**Motivation**

**Implications and formal contexts**

formal context $\leadsto$ few implications $\leadsto$ small formal context

How can be compute for a given set of implications a small corresponding formal context?

**Algorithms to compute the intents of a formal context** $(G, M, I)$

- NextClosure $\quad$ input: $M, X \mapsto X''$
- Close-by-One
Motivation

Implications and formal contexts

formal context $\leadsto$ few implications $\leadsto$ via NextClosure
implications $\leadsto$ small formal context $\leadsto$ ?

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context $(G, M, I)$

NextClosure input: $M, X \mapsto X''$
Close-by-One input: $(G, M, I)$
Motivation

Implications and formal contexts

formal context $\Rightarrow$ few implications via NextClosure
implications $\Rightarrow$ small formal context ?

How can be compute for a given set of implications a small corresponding formal context?

Algorithms to compute the intents of a formal context $(G, M, I)$

NextClosure input: $M, X \mapsto X''$
Close-by-One input: $(G, M, I)$

Can Close-by-One be applied to an arbitrary closure operator $c$?
Decomposing Closure Operators

Definition

Let $M$ be a finite set and let $c : \mathcal{P}(M) \rightarrow \mathcal{P}(M)$. Then $c$ is a closure operator on $M$ if and only if

- $c$ is monotone: $\forall A, B \subseteq M : A \subseteq B \implies c(A) \subseteq c(B)$,
- $c$ is extending: $\forall A \subseteq M : A \subseteq c(A)$,
- $c$ is idempotent: $\forall A \subseteq M : c(c(A)) = c(A)$.
Decomposing Closure Operators

Definition
Let $M$ be a finite set and let $c : \mathcal{P}(M) \to \mathcal{P}(M)$. Then $c$ is a closure operator on $M$ if and only if

- $c$ is monotone: $\forall A, B \subseteq M : A \subseteq B \implies c(A) \subseteq c(B)$,
- $c$ is extending: $\forall A \subseteq M : A \subseteq c(A)$,
- $c$ is idempotent: $\forall A \subseteq M : c(c(A)) = c(A)$.

Definition
A formal context $\mathbb{K} = (G, M, I)$ is a decomposition of $c$ if and only if

$$\text{Int}(\mathbb{K}) = c[\mathcal{P}(M)]$$

i.e., the intents of $\mathbb{K}$ are precisely the closed sets of $c$. 
Lemma

The formal context

\[ K_c = (c[\Psi(M)], M, \exists) \]

is a decomposition of \( c \).
Lemma

The formal context

\[ K_c = (c[\mathcal{P}(M)], M, \exists) \]

is a decomposition of \( c \).

Definition

The formal context \( K_c \) is called the \textit{trivial decomposition of} \( c \).
Lemma

Every object-clarified and object-reduced decomposition of a closure operator $c$ can be embedded into $K_c$. 
Lemma

Every object-clarified and object-reduced decomposition of a closure operator \( c \) can be embedded into \( K_c \).

\( K_c \) is therefore the biggest possible decomposition (up to object renaming).
Lemma

Every object-clarified and object-reduced decomposition of a closure operator $c$ can be embedded into $K_c$.

$K_c$ is therefore the biggest possible decomposition (up to object renaming).

But what about the smallest possible decomposition?
Definition

The *canonical decomposition of $c$* is the uniquely determined object-reduced subcontext of $K_c$. 

Lemma

The canonical decomposition of $c$ is the smallest possible decomposition of $c$, i.e., it can be embedded into every other decomposition of $c$. 

Can we compute the canonical decomposition without computing the trivial one?
Definition

The *canonical decomposition of c* is the uniquely determined object-reduced subcontext of $\mathbb{K}_c$.

Lemma

*The canonical decomposition of c is the smallest possible decomposition of c, i.e., it can be embedded into every other decomposition of c.*
Definition

The canonical decomposition of $c$ is the uniquely determined object-reduced subcontext of $K_c$.

Lemma

The canonical decomposition of $c$ is the smallest possible decomposition of $c$, i.e., it can be embedded into every other decomposition of $c$.

Can we compute the canonical decomposition without computing the trivial one?
Reconsidering the Problem

Given a closure operator $c$ on a set $M$. Then what do we have?
Reconsidering the Problem

Given a closure operator $c$ on a set $M$. Then what do we have?

- a set $M$ of attributes and
Reconsidering the Problem

Given a closure operator $c$ on a set $M$. Then what do we have?

- a set $M$ of attributes and
- the operator $X \mapsto X'' = c(X)$, computed in any decomposition of $c$. 
Given a closure operator \( c \) on a set \( M \). Then what do we have?
- a set \( M \) of attributes and
- the operator \( X \mapsto X'' = c(X) \), computed in any decomposition of \( c \).

We therefore have the logic of every decomposition of \( c \), i.e., we can decide whether an implication \( A \rightarrow B \) holds in a decomposition of \( c \) by checking

\[
B \subseteq c(A).
\]
Given a closure operator $c$ on a set $M$. Then what do we have?

- a set $M$ of attributes and
- the operator $X \mapsto X'' = c(X)$, computed in any decomposition of $c$.

We therefore have the logic of every decomposition of $c$, i.e., we can decide whether an implication $A \rightarrow B$ holds in a decomposition of $c$ by checking

$$B \subseteq c(A).$$

So can we do attribute exploration?
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$

- if $B \subseteq c(A)$ accept,
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$
- if $B \subseteq c(A)$ accept,
- otherwise deny.
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$

- if $B \subseteq c(A)$ accept,
- otherwise deny.

Then $A \rightarrow B$ does not respect $c(A)$
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$
- if $B \subseteq c(A)$ accept,
- otherwise deny.

Then $A \rightarrow B$ does not respect $c(A)$ and $c(A)$ is an intent of every decomposition of $c$. 
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$

- if $B \subseteq c(A)$ accept,
- otherwise deny.

Then $A \rightarrow B$ does not respect $c(A)$ and $c(A)$ is an intent of every decomposition of $c$.

Therefore provide $c(A)$ as a counterexample.
Turning the closure operator into an expert: Given an implication $A \rightarrow B$

- if $B \subseteq c(A)$ accept,
- otherwise deny.

Then $A \rightarrow B$ does not respect $c(A)$ and $c(A)$ is an intent of every decomposition of $c$.

Therefore provide $c(A)$ as a counterexample.

Now attribute exploration can be used to compute a decomposition of $c$!
Using Attribute Exploration

Turning the closure operator into an expert: Given an implication $A \rightarrow B$

- if $B \subseteq c(A)$ accept,
- otherwise deny.

Then $A \rightarrow B$ does not respect $c(A)$ and $c(A)$ is an intent of every decomposition of $c$.

Therefore provide $c(A)$ as a counterexample.

Now attribute exploration can be used to compute a decomposition of $c$!

But this will not always yield the canonical decomposition of $c$. 
Maximal Counterexamples

For an invalid implication $A \rightarrow B$, $c(A)$ is a counterexample, but it is not the only one.
Maximal Counterexamples

For an invalid implication $A \rightarrow B$, $c(A)$ is a counterexample, but it is not the only one.

*Every* closed set $C \supseteq A$ with $B \not\subseteq C$ is a counterexample for $A \rightarrow B$. 

**Lemma**

Let $N \in c[P(M)]$. Then $N$ is infimum-irreducible in $(c[P(M)], \subseteq)$ if and only if there exists an $n \in M \setminus N$ such that $N$ is maximal in $(c[P(M)], \subseteq)$ with respect to not containing $n$.

**Idea**

If $B \not\subseteq c(A)$, then choose $x \in B \setminus c(A)$ and maximize $N \supseteq c(A)$ with respect to $x \not\in N$. Then call $N$ a maximal counterexample for $A \rightarrow B$. 

Maximal Counterexamples

For an invalid implication \( A \rightarrow B \), \( c(A) \) is a counterexample, but it is not the only one.

Every closed set \( C \supseteq A \) with \( B \not\subseteq C \) is a counterexample for \( A \rightarrow B \).

**Lemma**

Let \( N \in c[\mathcal{P}(M)] \). Then \( N \) is infimum-irreducible in \((c[\mathcal{P}(M)], \subseteq)\) if and only if there exists an \( n \in M \setminus N \) such that \( N \) is maximal in \((c[\mathcal{P}(M)], \subseteq)\) with respect to not containing \( n \).
Maximal Counterexamples

For an invalid implication $A \rightarrow B$, $c(A)$ is a counterexample, but it is not the only one.

*Every* closed set $C \supseteq A$ with $B \nsubseteq C$ is a counterexample for $A \rightarrow B$.

**Lemma**

Let $N \in c[P(M)]$. Then $N$ is infimum-irreducible in $(c[P(M)], \subseteq)$ if and only if there exists an $n \in M \setminus N$ such that $N$ is maximal in $(c[P(M)], \subseteq)$ with respect to not containing $n$.

**Idea**

If $B \nsubseteq c(A)$, then choose $x \in B \setminus c(A)$ and maximize $N \supseteq c(A)$ with respect to $x \notin N$. 
Maximal Counterexamples

For an invalid implication $A \rightarrow B$, $c(A)$ is a counterexample, but it is not the only one.

*Every* closed set $C \supseteq A$ with $B \not\subseteq C$ is a counterexample for $A \rightarrow B$.

**Lemma**

Let $N \in c[\mathcal{P}(M)]$. Then $N$ is infimum-irreducible in $(c[\mathcal{P}(M)], \subseteq)$ if and only if there exists an $n \in M \setminus N$ such that $N$ is maximal in $(c[\mathcal{P}(M)], \subseteq)$ with respect to not containing $n$.

**Idea**

If $B \not\subseteq c(A)$, then choose $x \in B \setminus c(A)$ and maximize $N \supseteq c(A)$ with respect to $x \not\in N$. Then call $N$ a *maximal counterexample* for $A \rightarrow B$. 
Corollary

Attribute exploration using maximal counterexamples yields as the final context of the exploration the canonical decomposition of $c$. 
Fix $M := \{0, \ldots, 10\}$. 

Randomly generate formal contexts $K$ with attribute set $M$. 

Compute the canonical decomposition of $X \mapsto X''$ using the naive algorithm or attribute exploration with maximal counterexamples.
Experiments

- Fix $M := \{0, \ldots, 10\}$.
- Randomly generate formal contexts $\mathcal{K}$ with attribute set $M$. 
Fix $M := \{0, \ldots, 10\}$.
Randomly generate formal contexts $\mathbb{K}$ with attribute set $M$.
Compute the canonical decomposition of $X \mapsto X''$ using
Fix $M := \{0, \ldots, 10\}$.

Randomly generate formal contexts $\mathbb{K}$ with attribute set $M$.

Compute the canonical decomposition of $X \mapsto X''$ using

- the naive algorithm
Experiments

- Fix $M := \{0, \ldots, 10\}$.
- Randomly generate formal contexts $\mathcal{K}$ with attribute set $M$.
- Compute the canonical decomposition of $X \mapsto X''$ using
  - the naive algorithm
  - simple attribute exploration
Fix $M \equiv \{0, \ldots, 10\}$.

Randomly generate formal contexts $\mathcal{K}$ with attribute set $M$.

Compute the canonical decomposition of $X \mapsto X''$ using

- the naive algorithm
- simple attribute exploration
- attribute exploration with maximal counterexamples
Experimental Results

Number of intents vs. Runtime.
Experimental Results

Number of intents vs. Runtime.

Number of pseudo-intents vs. Runtime.
Experimental Results (cont.)

Calls of $c$ vs. Runtime.
An Unexpected Observation

Number of intents vs. Number of pseudo-intents.
Further Research

Open Questions

Complexity of decomposing closure operators?

Canonical decomposition might be exponentially large in $|M|$

How to represent $c$?

Correlation between number of intents and number of pseudo-intents?
Open Questions

- Complexity of decomposing closure operators?
Further Research

Open Questions

- Complexity of decomposing closure operators?
  - Canonical decomposition might be exponentially large in $|M|$
Further Research

Open Questions

- Complexity of decomposing closure operators?
  - Canonical decomposition might be exponentially large in $|M|$?
  - How to represent $c$?
Further Research

Open Questions

- Complexity of decomposing closure operators?
  - Canonical decomposition might be exponentially large in $|M|$
  - How to represent $c$?
- Correlation between number of intents and number of pseudo-intents?
Thank You.