

On the Complexity of Mining Itemsets from the Crowd Using Taxonomies

Antoine Amarilli^{1,2,3} Yael Amsterdamer¹ Tova Milo¹

¹Tel Aviv University, Tel Aviv, Israel

²École normale supérieure, Paris, France

³Télécom ParisTech, Paris, France



Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

$$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$

- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

- $$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$
- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
 - $\{\text{salad}\}$ not frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

- $$D = \left\{ \begin{array}{l} \{\text{beer}, \text{diapers}\}, \\ \{\text{beer}, \text{bread}, \text{butter}\}, \\ \{\text{beer}, \text{bread}, \text{diapers}\}, \\ \{\text{salad}, \text{tomato}\} \end{array} \right\}$$
- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
 - **{salad}** not frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

- $$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$
- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
 - $\{\text{salad}\}$ not frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

- $$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$
- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
 - $\{\text{salad}\}$ not frequent
 - $\{\text{beer, diapers}\}$ frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

$$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$

- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\text{salad}\}$ not frequent
- $\{\text{beer, diapers}\}$ frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

$$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$

- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\text{salad}\}$ not frequent
- $\{\text{beer, diapers}\}$ frequent
 \Rightarrow $\{\text{beer}\}$ is also frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

$$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$

- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\text{salad}\}$ not frequent
- $\{\text{beer, diapers}\}$ frequent
 \Rightarrow $\{\text{beer}\}$ is also frequent

Frequent itemset mining

Data mining – discovering interesting patterns in large **databases**

Database – a (multi)set of **transactions**

Transaction – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

$$D = \left\{ \begin{array}{l} \{\text{beer, diapers}\}, \\ \{\text{beer, bread, butter}\}, \\ \{\text{beer, bread, diapers}\}, \\ \{\text{salad, tomato}\} \end{array} \right\}$$

- Itemset is **frequent** if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\text{salad}\}$ not frequent
- $\{\text{beer, diapers}\}$ frequent
 \Rightarrow $\{\text{beer}\}$ is also frequent

Human knowledge mining

- Some databases only exist in the **minds** of people
- Example: **popular activities in Athens**:
 - t_1 : I went to the **acropolis** and then to the **museum**.
⇒ {acropolis, museum}
 - t_2 : I visited **Piraeus** and had some **ice cream**.
⇒ {piraeus, icecream}
 - t_3 : On Monday I attended the **keynote** and then had **coffee**.
⇒ {keynote, coffee}

Human knowledge mining

- Some databases only exist in the **minds** of people
 - Example: **popular activities in Athens**:
 - t_1 : I went to the **acropolis** and then to the **museum**.
⇒ {acropolis, museum}
 - t_2 : I visited **Piraeus** and had some **ice cream**.
⇒ {piraeus, icecream}
 - t_3 : On Monday I attended the **keynote** and then had **coffee**.
⇒ {keynote, coffee}
 - We want **frequent itemsets**: frequent **activity combinations**
- ⇒ How to **retrieve** this data from people?

Harvesting the data

- We **cannot** collect such data in a centralized database:
 - ① It's **impractical** to ask all users to surrender their data
"Everyone please tell us all you did the last three months."
 - ② People do not **remember** the information
"What were you doing on August 23th, 2013?"

Harvesting the data

- We **cannot** collect such data in a centralized database:

- ① It's **impractical** to ask all users to surrender their data

“Everyone please tell us all you did the last three months.”

- ② People do not **remember** the information

“What were you doing on August 23th, 2013?”

- People remember **summaries** that we could access

“Do you often eat ice cream when attending a keynote?”

⇒ We can just **ask** people if an itemset is frequent

Crowdsourcing

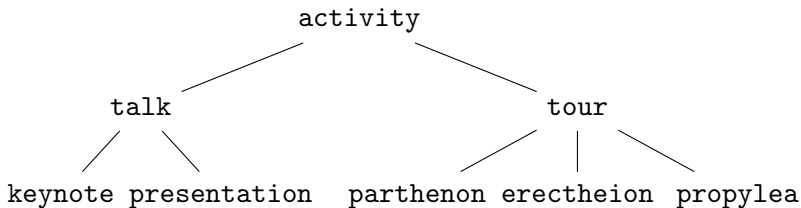
- **Crowdsourcing** – solving hard problems through elementary queries to a crowd of users
- Find out if an itemset is **frequent** with the crowd:
 - ① **Draw** a sample of users from the crowd. *(black box)*
 - ② **Ask**: is this itemset frequent? *(“Do you often have coffee?”)*
 - ③ **Corroborate** the answers to eliminate bad answers. *(black box)*
 - ④ **Reward** the users. *(e.g., monetary incentive)*

Crowdsourcing

- **Crowdsourcing** – solving hard problems through elementary queries to a crowd of users
 - Find out if an itemset is **frequent** with the crowd:
 - ① **Draw** a sample of users from the crowd. (*black box*)
 - ② **Ask**: is this itemset frequent? (*“Do you often have coffee?”*)
 - ③ **Corroborate** the answers to eliminate bad answers. (*black box*)
 - ④ **Reward** the users. (*e.g., monetary incentive*)
- ⇒ The crowd is an **oracle**: given an itemset, say if it is frequent

Taxonomies

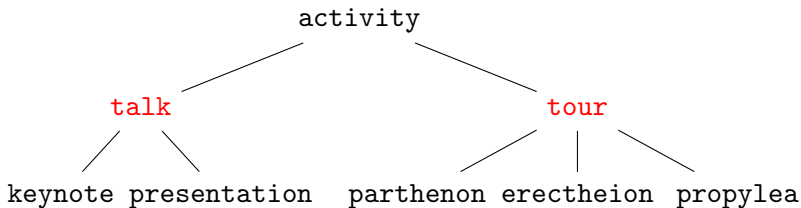
Having a **taxonomy** over the items can save us work!



- {talk, tour} infrequent

Taxonomies

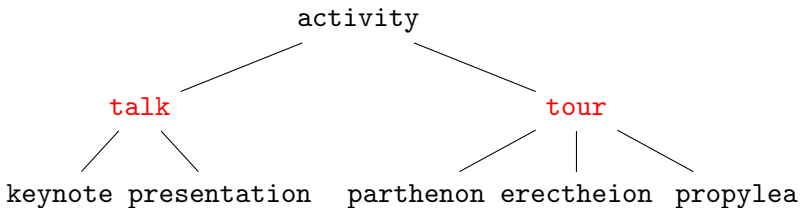
Having a **taxonomy** over the items can save us work!



- {talk, tour} infrequent

Taxonomies

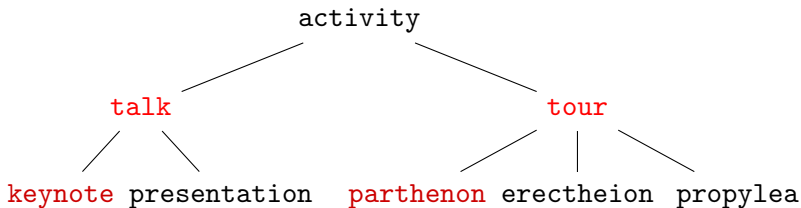
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent

Taxonomies

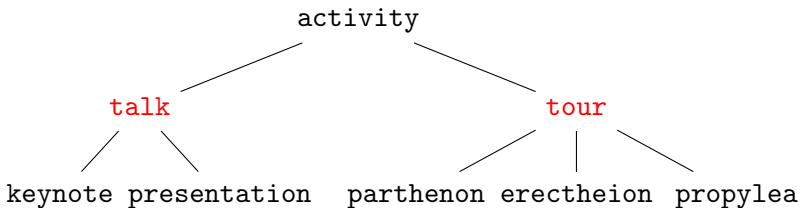
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as **{keynote, parthenon}** also infrequent

Taxonomies

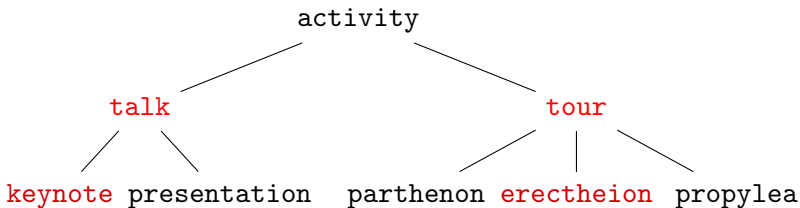
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

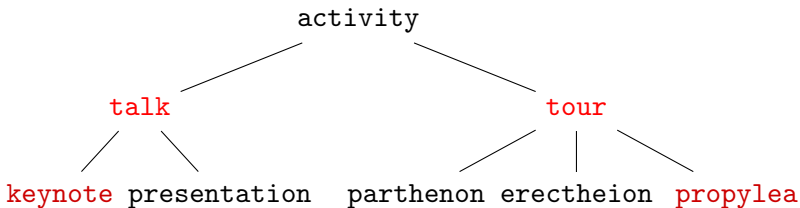
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

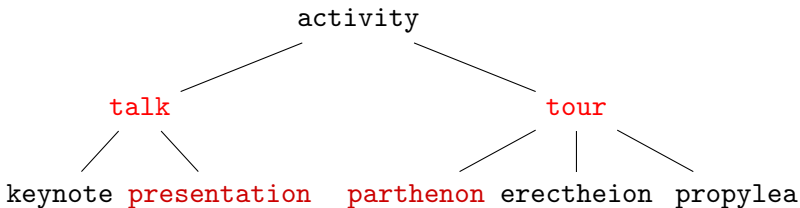
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

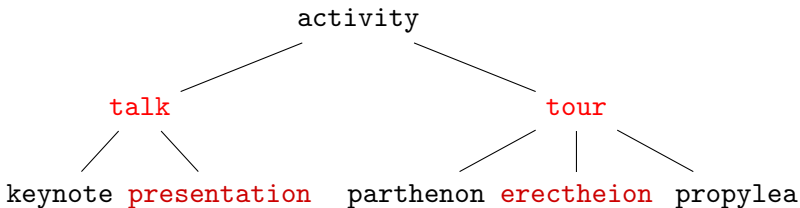
Having a **taxonomy** over the items can save us work!



- **{talk, tour}** infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

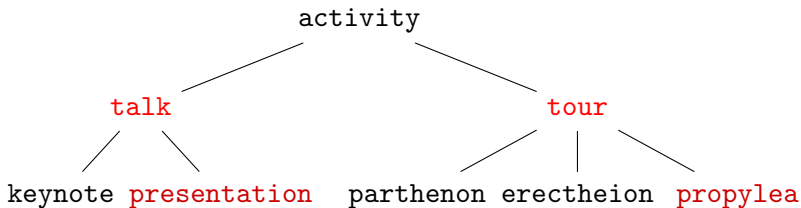
Having a **taxonomy** over the items can save us work!



- {**talk, tour**} infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

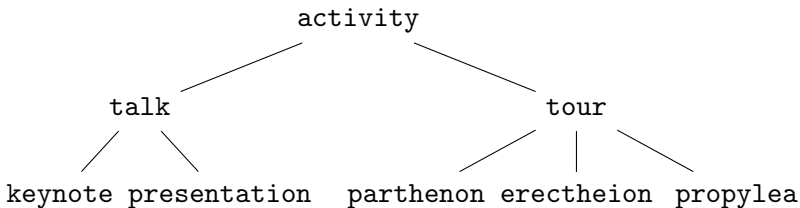
Having a **taxonomy** over the items can save us work!



- {**talk**, **tour**} infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**

Taxonomies

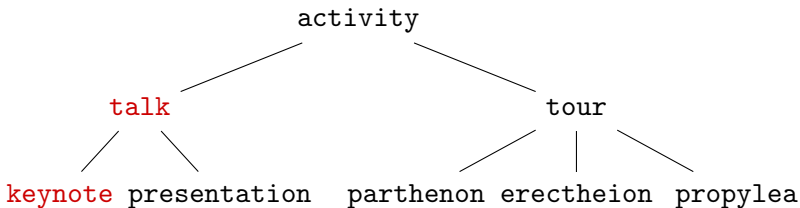
Having a **taxonomy** over the items can save us work!



- {talk, tour} infrequent
 - ⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**
- Also avoids **redundant itemsets** like {talk, keynote}

Taxonomies

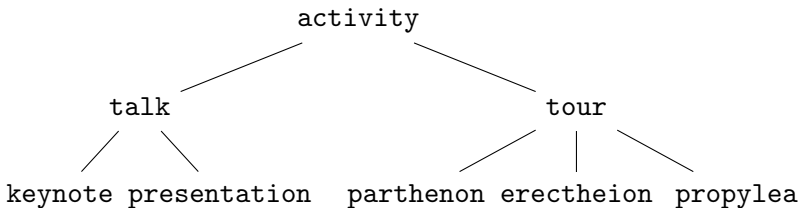
Having a **taxonomy** over the items can save us work!



- {talk, tour} infrequent
⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**
- Also avoids **redundant itemsets** like {talk, keynote}

Taxonomies

Having a **taxonomy** over the items can save us work!



- {talk, tour} infrequent
 - ⇒ Itemsets such as {keynote, parthenon} also infrequent
- Without the taxonomy, we need to test **all combinations!**
- Also avoids **redundant itemsets** like {talk, keynote}

The problem

We can now describe the **problem**:

- We have:
 - A known **item domain** \mathcal{I} (set of items)
 - A known **taxonomy** Ψ on \mathcal{I} (is-a relation, partial order)
 - A crowd **oracle** to decide if an itemset is frequent or not
 - Choose questions **interactively** based on past answers
- ⇒ Find out the status of **all** itemsets

The problem

We can now describe the **problem**:

- We have:
 - A known **item domain** \mathcal{I} (set of items)
 - A known **taxonomy** Ψ on \mathcal{I} (is-a relation, partial order)
 - A crowd **oracle** to decide if an itemset is frequent or not
 - Choose questions **interactively** based on past answers
- ⇒ Find out the status of **all** itemsets

What is a good algorithm to solve this problem?

Cost

- How to evaluate the **performance** of a strategy to identify the frequent itemsets?
Crowd complexity: The number of itemsets we ask about (monetary cost, latency...)
Computational complexity: The complexity of computing the next question to ask

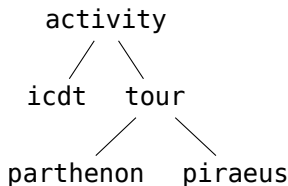
Cost

- How to evaluate the **performance** of a strategy to identify the frequent itemsets?
 - Crowd complexity:** The number of itemsets we ask about (monetary cost, latency...)
 - Computational complexity:** The complexity of computing the next question to ask
- **Tradeoff** between the two:
 - ⇒ Asking **random** questions: computationally inexpensive but bad crowd complexity
 - ⇒ Asking **clever** questions: optimal crowd complexity but computationally expensive

Table of contents

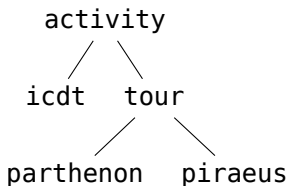
- 1 Background
- 2 Preliminaries**
- 3 Crowd complexity
- 4 Output crowd complexity
- 5 Computational complexity
- 6 Conclusion

Itemsets



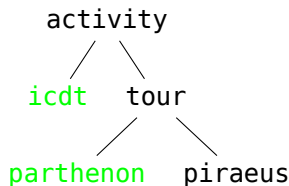
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items

Itemsets



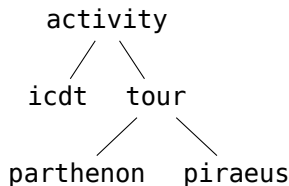
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset

Itemsets



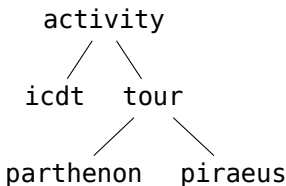
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset

Itemsets



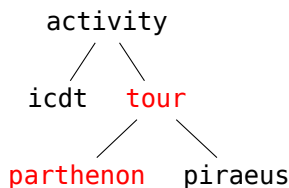
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset

Itemsets



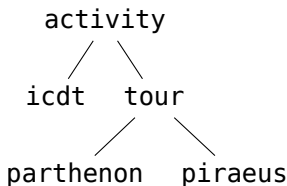
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not

Itemsets



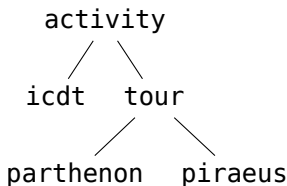
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not

Itemsets



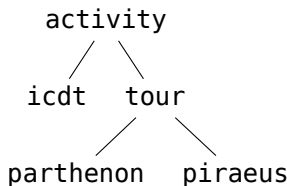
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not

Itemsets



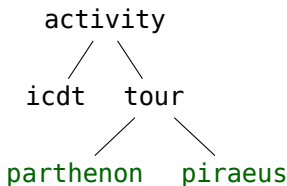
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:

Itemsets



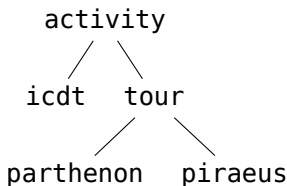
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent

Itemsets



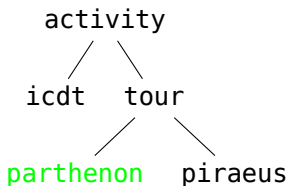
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent

Itemsets



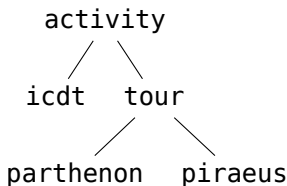
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent

Itemsets



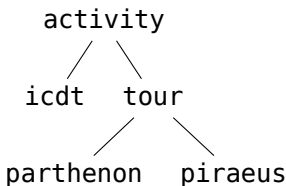
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent

Itemsets



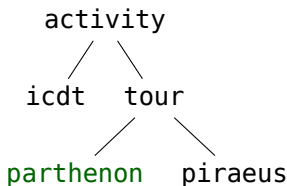
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent

Itemsets



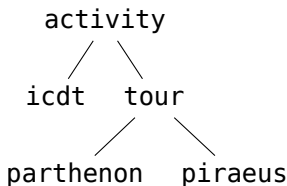
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent
 - $\{\text{parthenon}\}$ frequent

Itemsets



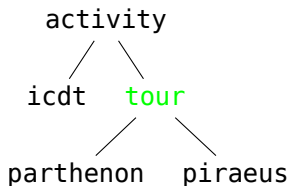
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent
 - $\{\text{parthenon}\}$ frequent

Itemsets



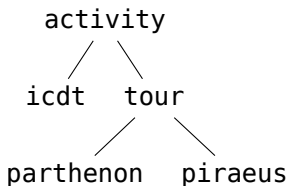
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent
 - $\{\text{parthenon}\}$ frequent
 - ⇒ $\{\text{tour}\}$ also frequent

Itemsets



- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent
 - $\{\text{parthenon}\}$ frequent
 - ⇒ $\{\text{tour}\}$ also frequent

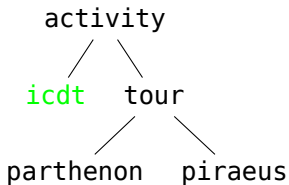
Itemsets



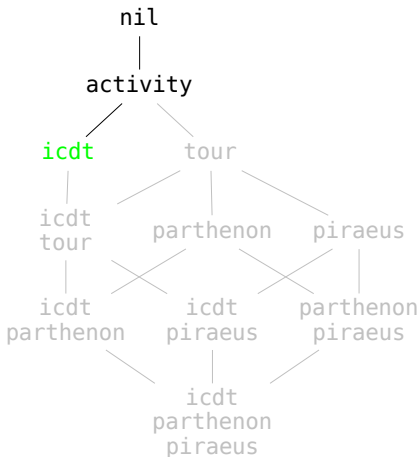
- **Itemsets** $I(\Psi)$ – the sets of **pairwise incomparable** items
 - ⇒ $\{\text{icdt}, \text{parthenon}\}$ is an itemset
 - ⇒ $\{\text{tour}, \text{parthenon}\}$ is not
- **Order** over itemsets:
 - $\{\text{parthenon}, \text{piraeus}\}$ frequent
 - ⇒ $\{\text{parthenon}\}$ also frequent
 - $\{\text{parthenon}\}$ frequent
 - ⇒ $\{\text{tour}\}$ also frequent

Itemset taxonomy example

Taxonomy Ψ

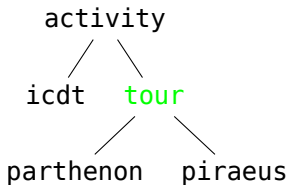


Itemset taxonomy $I(\Psi)$

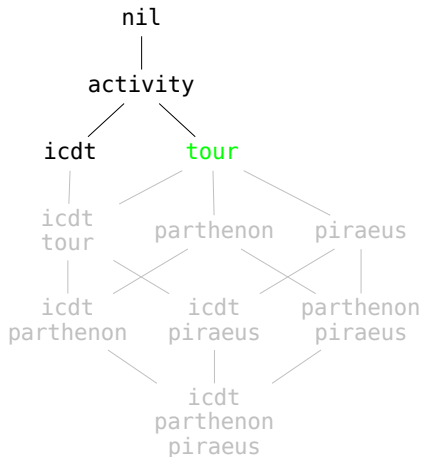


Itemset taxonomy example

Taxonomy Ψ

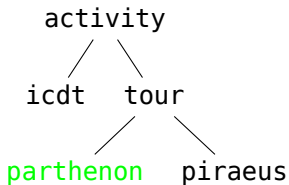


Itemset taxonomy $I(\Psi)$

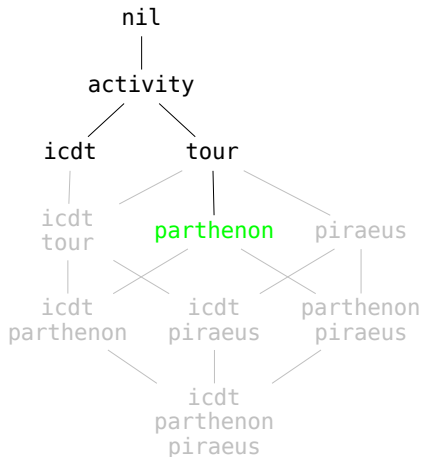


Itemset taxonomy example

Taxonomy Ψ

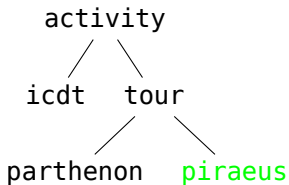


Itemset taxonomy $I(\Psi)$

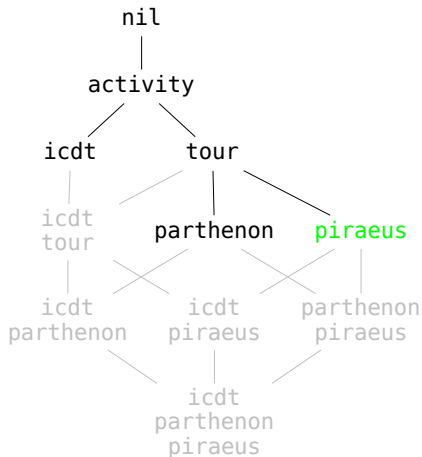


Itemset taxonomy example

Taxonomy Ψ

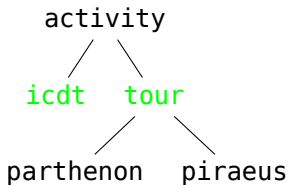


Itemset taxonomy $I(\Psi)$

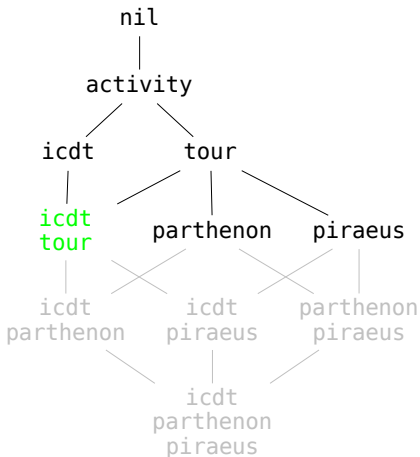


Itemset taxonomy example

Taxonomy Ψ

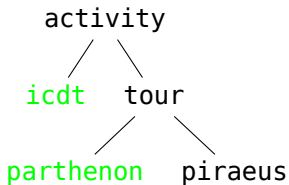


Itemset taxonomy $I(\Psi)$

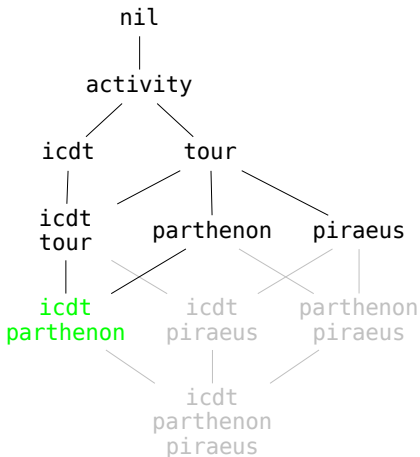


Itemset taxonomy example

Taxonomy Ψ

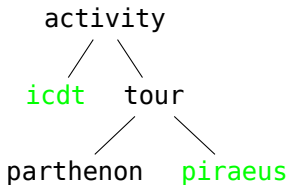


Itemset taxonomy $I(\Psi)$

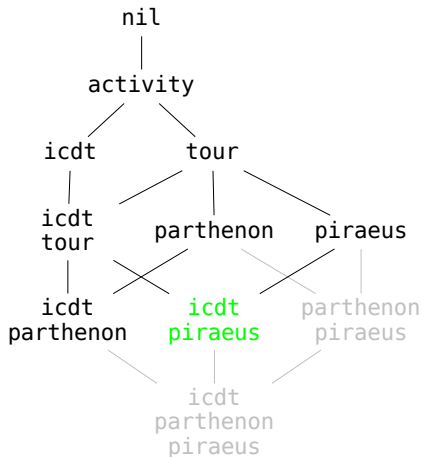


Itemset taxonomy example

Taxonomy Ψ

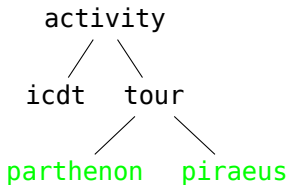


Itemset taxonomy $I(\Psi)$

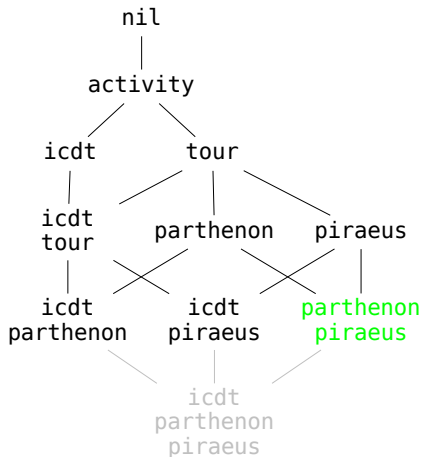


Itemset taxonomy example

Taxonomy Ψ

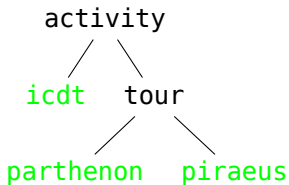


Itemset taxonomy $I(\Psi)$

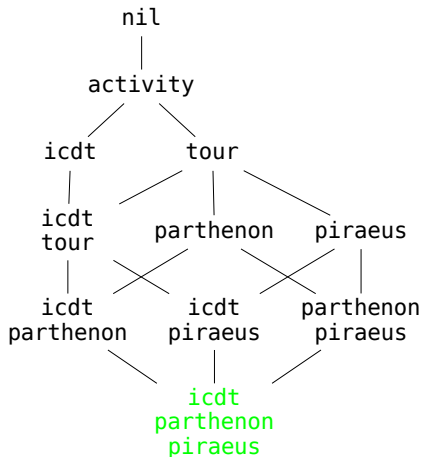


Itemset taxonomy example

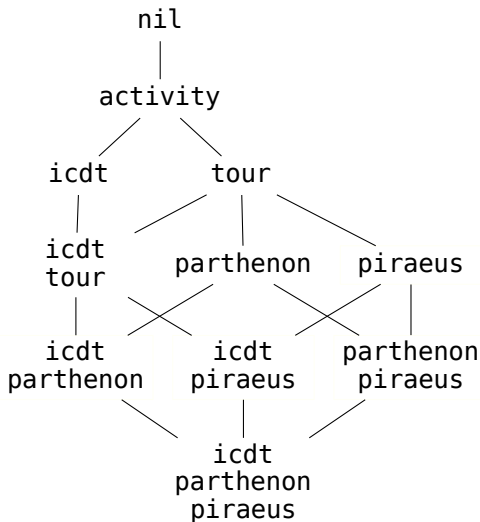
Taxonomy Ψ



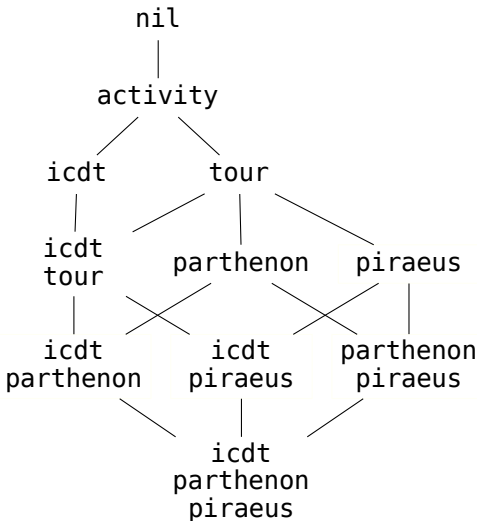
Itemset taxonomy $I(\Psi)$



Solutions

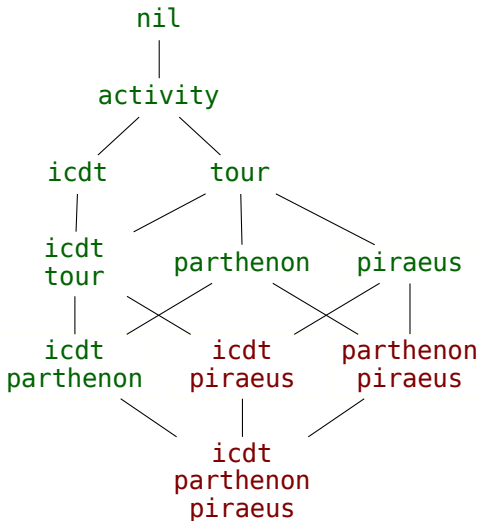


Solutions



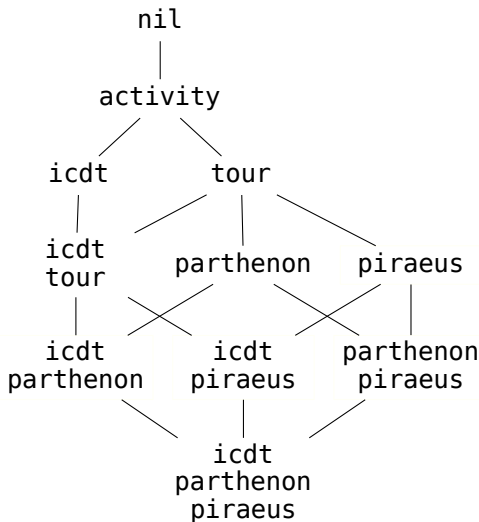
- “Being frequent” is a **monotone** predicate over $I(\Psi)$

Solutions



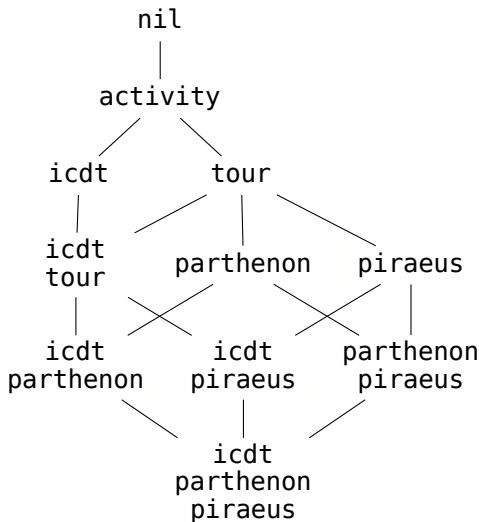
- “Being frequent” is a **monotone** predicate over $I(\Psi)$

Solutions



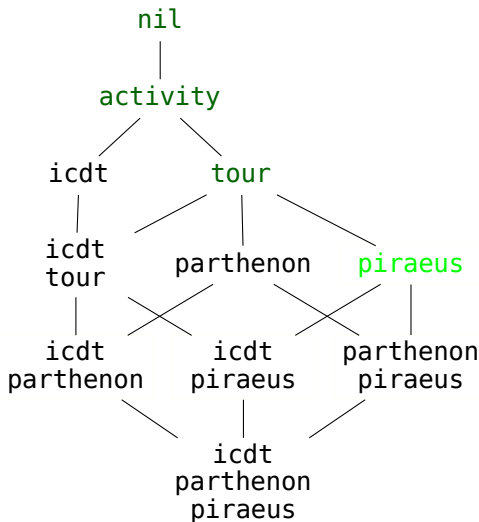
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:

Solutions



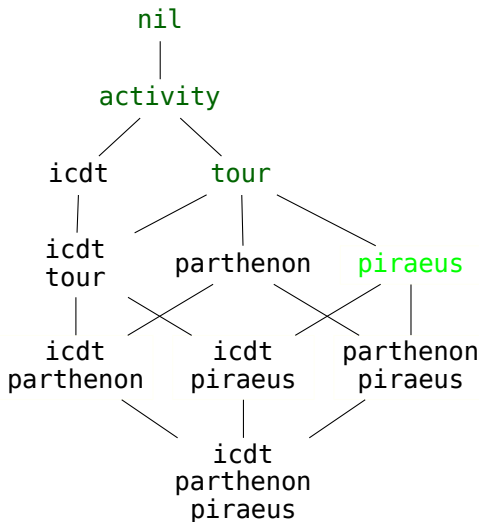
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is {piraeus} frequent?

Solutions



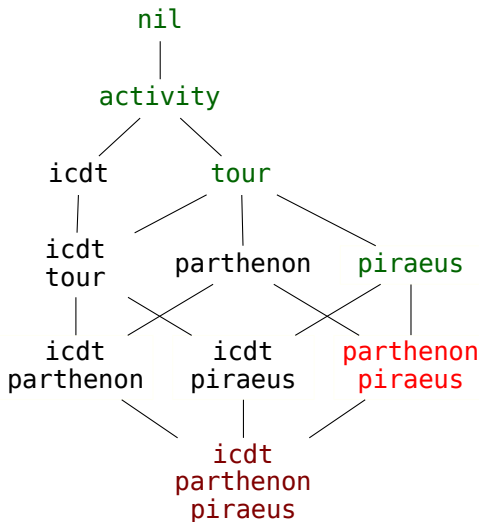
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is **{piraeus}** frequent?
⇒ **Yes!**

Solutions



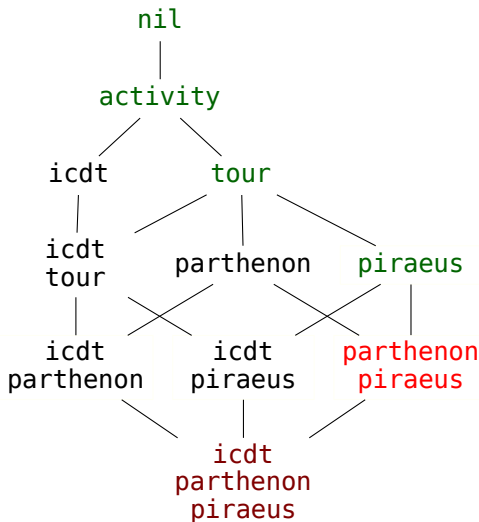
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?

Solutions



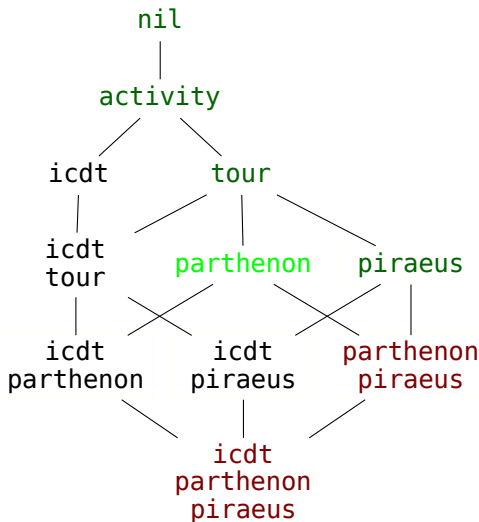
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?
⇒ No!

Solutions



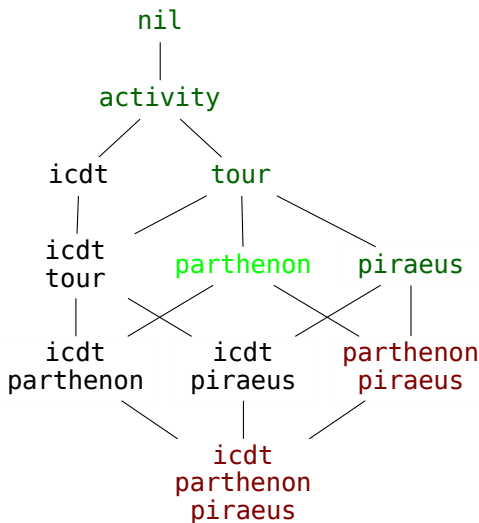
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?
⇒ No!
- $\{\text{parthenon}\}$?

Solutions



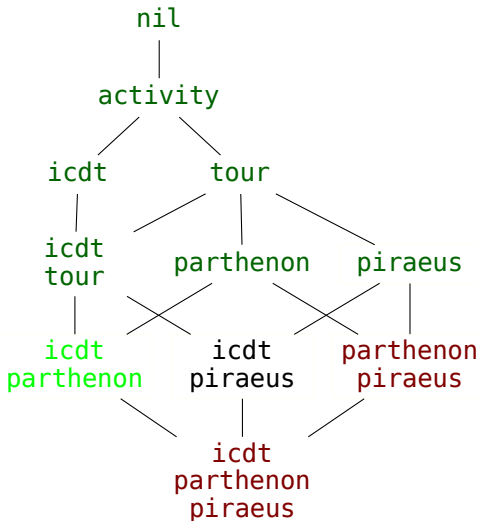
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?
⇒ No!
- $\{\text{parthenon}\}$?
⇒ Yes!

Solutions



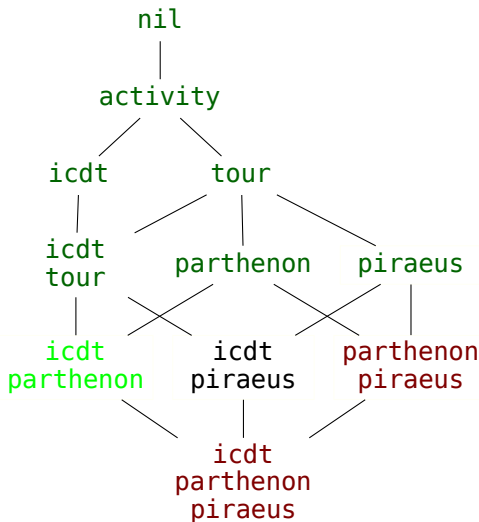
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is {piraeus} frequent?
⇒ Yes!
- {parthenon, piraeus}?
⇒ No!
- {parthenon}?
⇒ Yes!
- {icdt, parthenon}?

Solutions



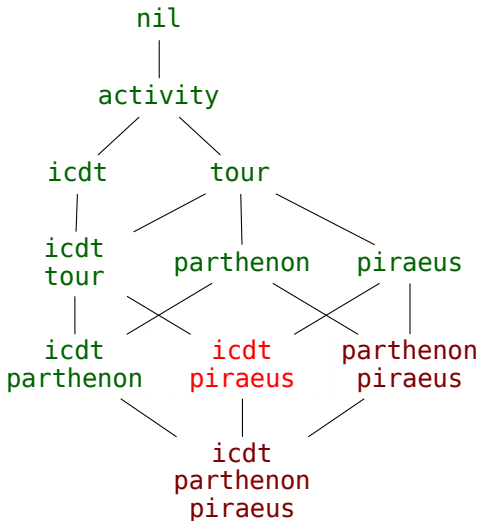
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?
⇒ No!
- $\{\text{parthenon}\}$?
⇒ Yes!
- $\{\text{icdt}, \text{parthenon}\}$?
⇒ Yes!

Solutions



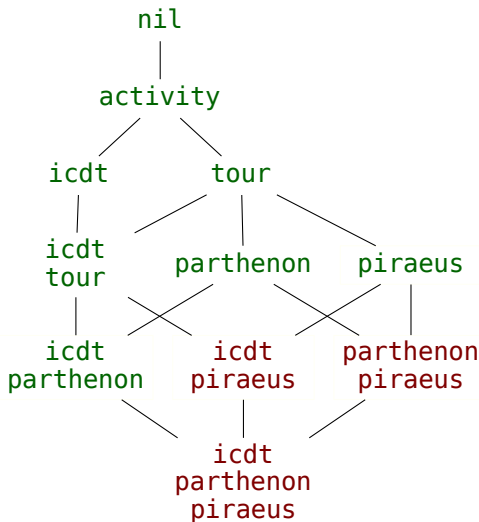
- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is $\{\text{piraeus}\}$ frequent?
⇒ Yes!
- $\{\text{parthenon}, \text{piraeus}\}$?
⇒ No!
- $\{\text{parthenon}\}$?
⇒ Yes!
- $\{\text{icdt}, \text{parthenon}\}$?
⇒ Yes!
- $\{\text{icdt}, \text{piraeus}\}$?

Solutions



- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is {piraeus} frequent?
⇒ Yes!
- {parthenon, piraeus}?
⇒ No!
- {parthenon}?
⇒ Yes!
- {icdt, parthenon}?
⇒ Yes!
- {icdt, piraeus}?
⇒ No!

Solutions



- “Being frequent” is a **monotone** predicate over $I(\Psi)$
- Ask **questions**:
- Is {piraeus} frequent?
⇒ Yes!
- {parthenon, piraeus}?
⇒ No!
- {parthenon}?
⇒ Yes!
- {icdt, parthenon}?
⇒ Yes!
- {icdt, piraeus}?
⇒ No!

Table of contents

- 1 Background
- 2 Preliminaries
- 3 Crowd complexity**
- 4 Output crowd complexity
- 5 Computational complexity
- 6 Conclusion

Crowd complexity lower bound

- How many **questions** do we need to ask?
- Each query yields **one bit** of information

Crowd complexity lower bound

- How many **questions** do we need to ask?
- Each query yields **one bit** of information
- **Information-theoretic lower bound**: at least $\Omega(\log N)$ queries, with N the number of solutions
- $N = \Omega(2^{|\Psi|})$ and $|\Psi| = \Omega(2^{|\Psi|})$
- W.r.t. the **original taxonomy** Ψ , $\Omega\left(2^{\text{width}(\Psi)} / \sqrt{\text{width}[\Psi]}\right)$

Crowd complexity upper bound

nil	6/7
a1	5/7
a2	4/7
a3	3/7
a4	2/7
a5	1/7

- Query itemsets that are frequent in **about half** of the solutions

Crowd complexity upper bound

nil	6/7
a1	5/7
a2	4/7
a3	3/7
a4	2/7
a5	1/7

- Query itemsets that are frequent in **about half** of the solutions

Crowd complexity upper bound

nil	6/7
a1	5/7
a2	4/7
a3	3/7
a4	2/7
a5	1/7

- Query itemsets that are frequent in **about half** of the solutions
- Itemset **split**: min of proportion where frequent and proportion where infrequent

Crowd complexity upper bound

nil	6/7	1/7
a1	5/7	2/7
a2	4/7	3/7
a3	3/7	3/7
a4	2/7	2/7
a5	1/7	1/7

- Query itemsets that are frequent in **about half** of the solutions
- Itemset **split**: min of proportion where frequent and proportion where infrequent

Crowd complexity upper bound

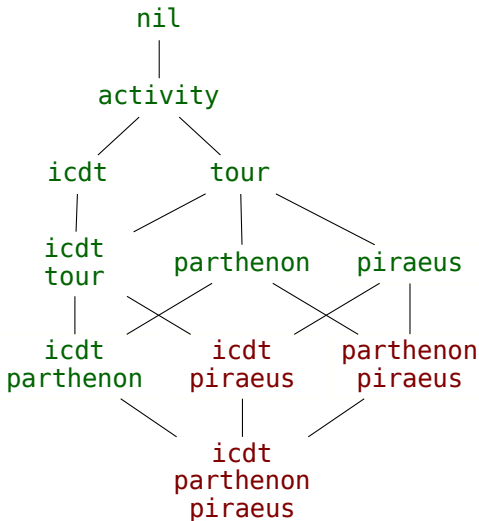
nil	6/7	1/7
a1	5/7	2/7
a2	4/7	3/7
a3	3/7	3/7
a4	2/7	2/7
a5	1/7	1/7

- Query itemsets that are frequent in **about half** of the solutions
 - Itemset **split**: min of proportion where frequent and proportion where infrequent
 - Existing result from **order theory** [Linial and Saks, 1985]: there is a **constant** $\delta_0 \approx 1/5$ such that some itemset achieves a split $\geq \delta_0$
- ⇒ The previous bound is **tight**: we need $\Theta(\log N)$ queries

Table of contents

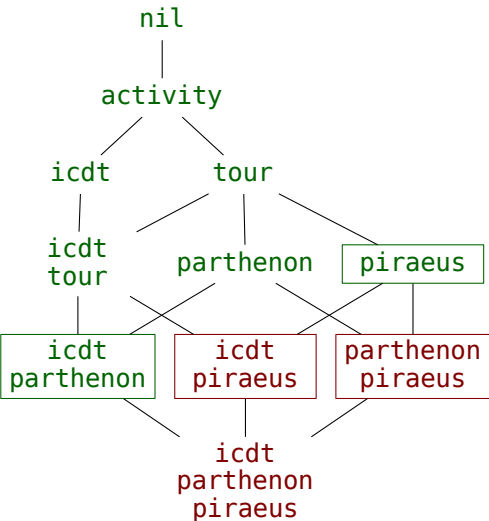
- 1 Background
- 2 Preliminaries
- 3 Crowd complexity
- 4 Output crowd complexity**
- 5 Computational complexity
- 6 Conclusion

Maximal frequent itemsets



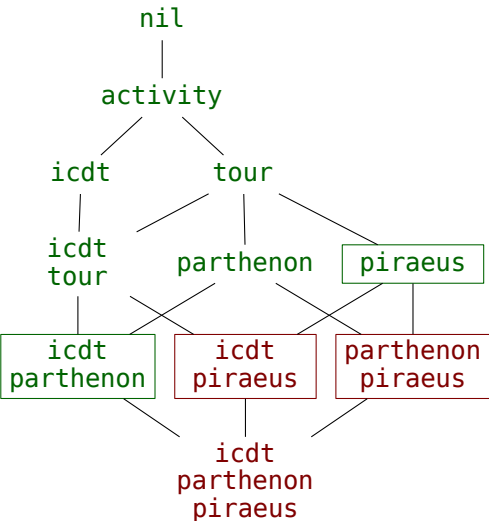
- Complexity with respect to the **output size**
- Output representation: **Maximal frequent itemsets (MFI)**
- **Minimal infrequent itemset (MII)**

Maximal frequent itemsets



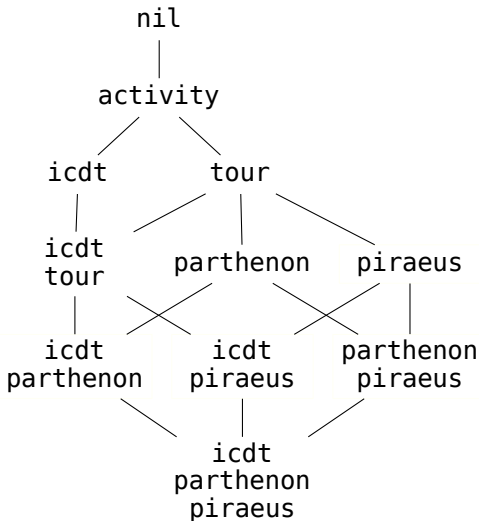
- Complexity with respect to the **output size**
- Output representation: **Maximal frequent itemsets (MFI)**
- **Minimal infrequent itemset (MII)**

Maximal frequent itemsets



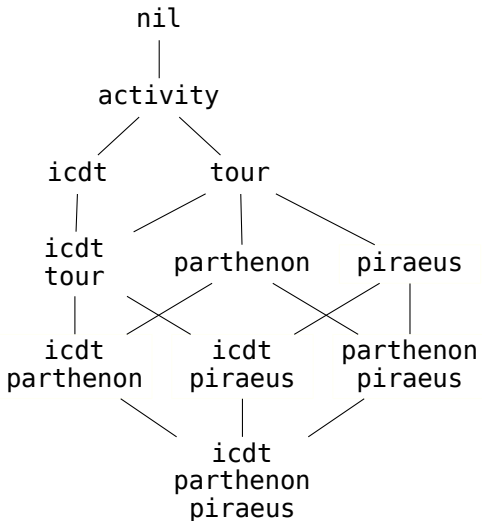
- Complexity with respect to the **output size**
- Output representation: **Maximal frequent itemsets** (MFI)
- **Minimal infrequent itemset** (MII)
- Must query **all** MFIs and MIIs
- Solutions with few MFIs/MIIs should be **easier to find**

MFI/MII upper bound



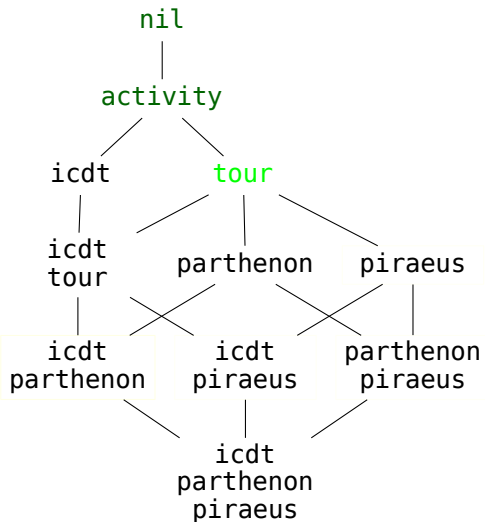
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**

MFI/MII upper bound



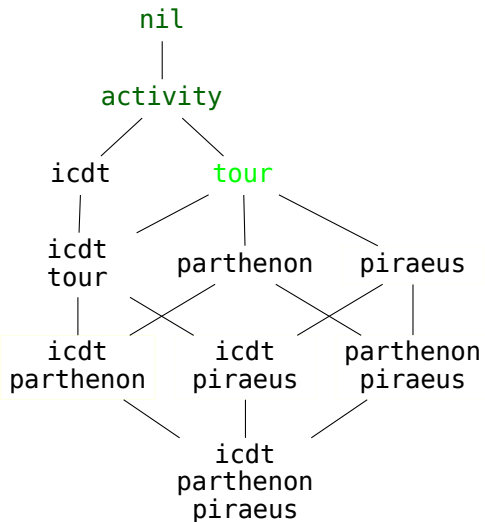
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**:

MFI/MII upper bound



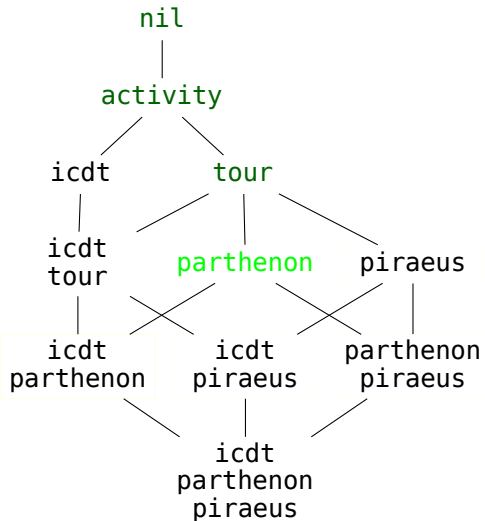
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: `{tour}`

MFI/MII upper bound



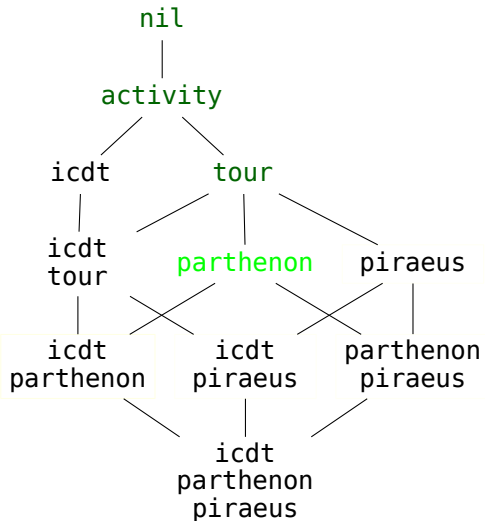
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...

MFI/MII upper bound



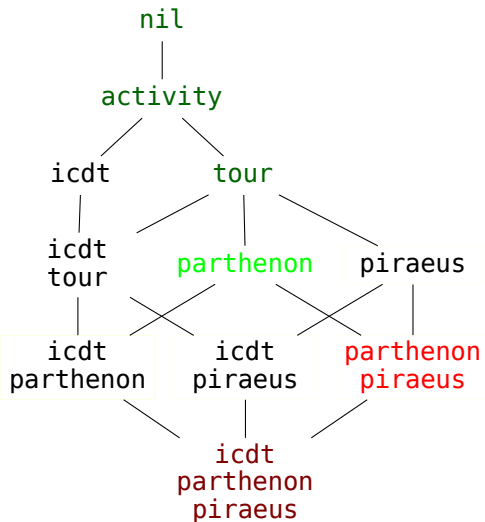
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...

MFI/MII upper bound



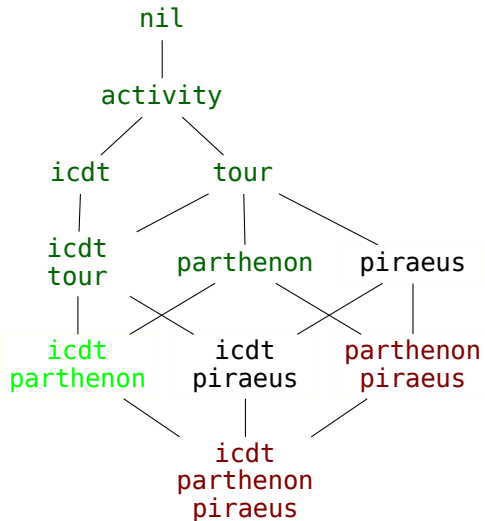
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can

MFI/MII upper bound



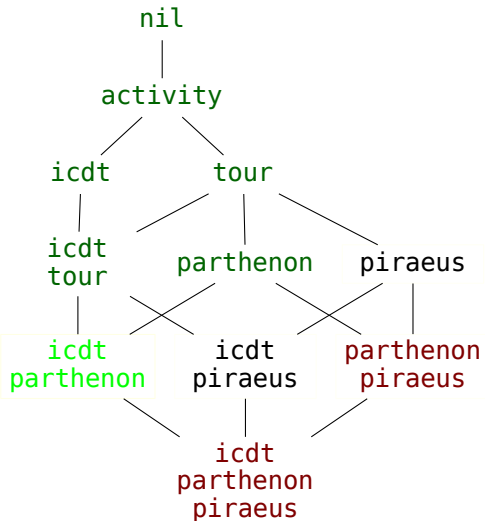
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can

MFI/MII upper bound



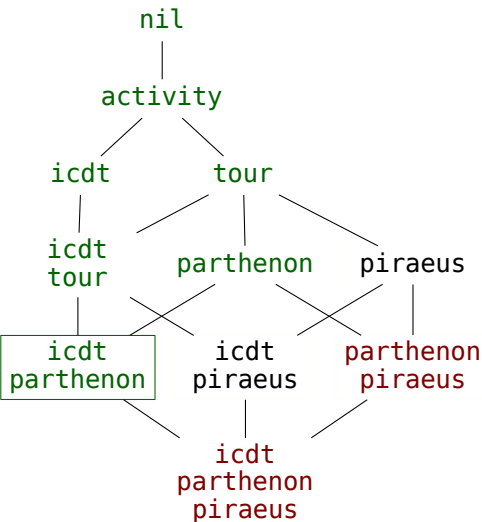
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can

MFI/MII upper bound



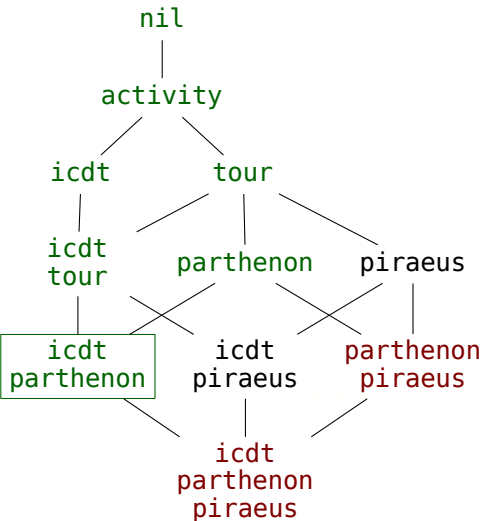
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can
 - Reach an MFI/MII

MFI/MII upper bound



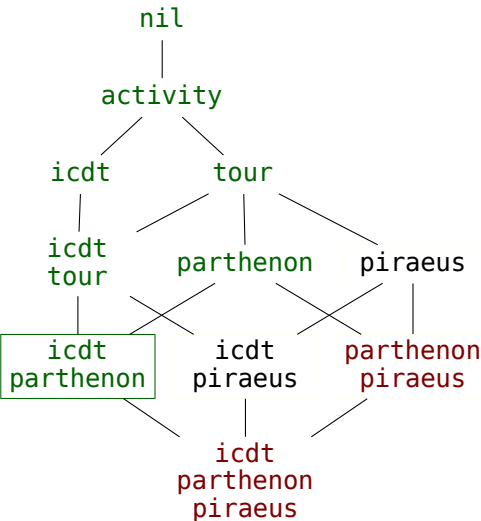
- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can
 - Reach an MFI/MII

MFI/MII upper bound



- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries
- **Example:**
 - Pick an **itemset**: {tour}
 - Specialize it...
 - ... while you can
 - Reach an MFI/MII
- At most $|\mathcal{I}|$ specializations

MFI/MII upper bound



- **Explicit algorithm** to find each MFI/MII in $\leq |\mathcal{I}|$ queries

- **Example:**

- Pick an **itemset**: {tour}
- Specialize it...
- ... while you can
- Reach an MFI/MII

- At most $|\mathcal{I}|$ specializations

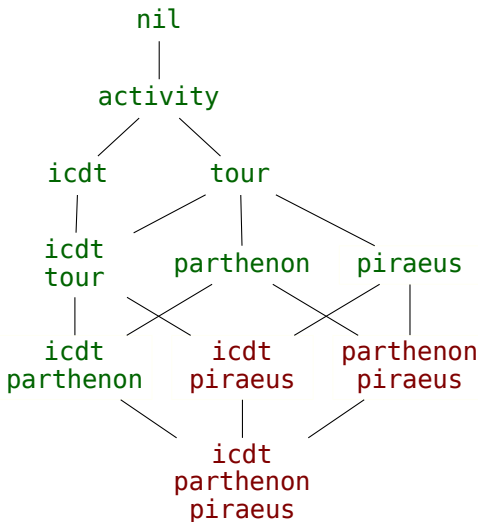
⇒ **Complexity:**

$$O(|\mathcal{I}| \cdot (|\text{MFI}| + |\text{MII}|))$$

Table of contents

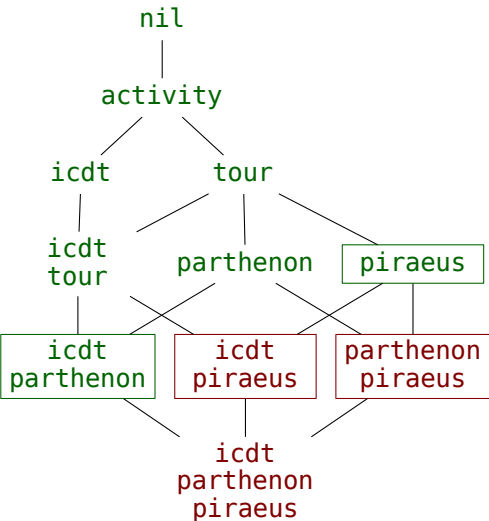
- 1 Background
- 2 Preliminaries
- 3 Crowd complexity
- 4 Output crowd complexity
- 5 Computational complexity**
- 6 Conclusion

Output computational complexity lower bound



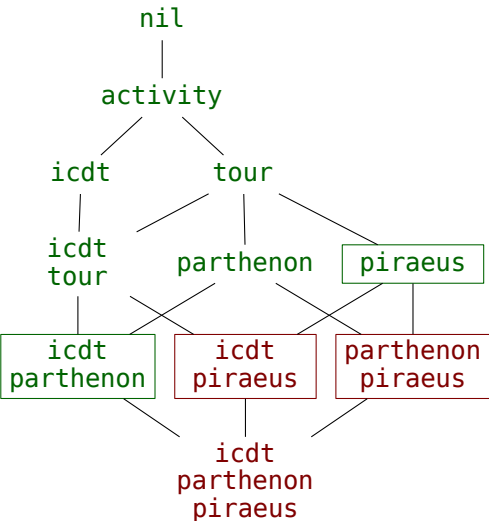
- Previous algorithm assumes $|\Psi|$ is **materialized**
- Do we **need** to?
- Decide if **finished**: do the MFIs/MIIIs cover all itemsets?

Output computational complexity lower bound



- Previous algorithm assumes $|I(\Psi)|$ is **materialized**
- Do we **need** to?
- Decide if **finished**: do the MFIs/MIIIs cover all itemsets?

Output computational complexity lower bound



- Previous algorithm assumes $|I(\Psi)|$ is **materialized**
- Do we **need** to?
- Decide if **finished**: do the MFIs/MILs cover all itemsets?
- This is **EQ-hard**, for problem EQ [Bioch and Ibaraki, 1995] (exact complexity open)

Computational complexity lower bound

- Find an unclassified itemset of $I(\Psi)$ frequent for **about half** of the possible solutions
 - We can **count** the possible solutions (exponential in $|I(\Psi)|$)
 - A solution is an “itemset” of $I(\Psi)$, an **antichain**, and counting the **antichains** of $I(\Psi)$ is $FP^{\#P}$ -complete.
- ⇒ Finding the best-split element in $I(\Psi)$ is **$FP^{\#P}$ -hard** in $|I(\Psi)|$?

Computational complexity lower bound

- Find an unclassified itemset of $I(\Psi)$ frequent for **about half** of the possible solutions
 - We can **count** the possible solutions (exponential in $|I(\Psi)|$)
 - A solution is an “itemset” of $I(\Psi)$, an **antichain**, and counting the **antichains** of $I(\Psi)$ is $FP^{\#P}$ -complete.
- ⇒ Finding the best-split element in $I(\Psi)$ is **$FP^{\#P}$ -hard** in $|I(\Psi)|$?
- **Problem:** $I(\Psi)$ is not a general DAG, so we only show hardness in $|\Psi|$ for restricted (fixed-size) itemsets
 - **Intuition:** count antichains by comparing to a known poset; use a best-split oracle to compare; perform a binary search

Table of contents

- 1 Background
- 2 Preliminaries
- 3 Crowd complexity
- 4 Output crowd complexity
- 5 Computational complexity
- 6 Conclusion**

Summary and further work

- **Problem:** mine frequent itemsets with the crowd
- Balance **crowd complexity** and **computational complexity**
- Function of the **input taxonomy size** or the **output size**

Summary and further work

- **Problem:** mine frequent itemsets with the crowd
- Balance **crowd complexity** and **computational complexity**
- Function of the **input taxonomy size** or the **output size**
- Future work:
 - Improve the **bounds** and close gaps
 - Benchmark **heuristics** (chain partitioning, random, etc.)
 - Manage **uncertainty** (black box for now)
 - Focus on **top- k** itemsets (work in progress)

Summary and further work

- **Problem:** mine frequent itemsets with the crowd
- Balance **crowd complexity** and **computational complexity**
- Function of the **input taxonomy size** or the **output size**
- Future work:
 - Improve the **bounds** and close gaps
 - Benchmark **heuristics** (chain partitioning, random, etc.)
 - Manage **uncertainty** (black box for now)
 - Focus on **top- k** itemsets (work in progress)

Thanks for your attention!

References



Bioch, J. and Ibaraki, T. (1995).

Complexity of identification and dualization of positive Boolean functions.

Inf. Comput., 123(1).



Linial, N. and Saks, M. (1985).

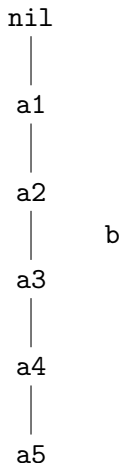
Every poset has a central element.

J. Combinatorial Theory, 40(2).

Greedy algorithms

- Querying an element of the chain may remove $< 1/2$ possible solutions
 - Querying the isolated element b will remove exactly $1/2$ solution
 - However, querying b classifies far less itemsets
- ⇒ Classifying many itemsets isn't the same as eliminating many solutions

Finding the greedy-best-split item is $\text{FP}^{\#P}$ -hard



Restricted itemsets

- Asking about **large** itemsets is irrelevant.

“Do you often go cycling and running while drinking coffee and having lunch with orange juice on alternate Wednesdays?”

- If the itemset size is bounded by a **constant**, $I(\Psi)$ is tractable
- ⇒ The crowd complexity $\Theta(\log |S(\Psi)|)$ is **tractable** too

Chain partitioning

- Optimal strategy for **chain taxonomies**: binary search
- We can determine a **chain decomposition** of the itemset taxonomy and perform binary searches on the chains
- **Optimal** crowd complexity for a chain, performance in general is unclear
- Computational complexity is **polynomial** in the size of $I(\Psi)$ (which is still exponential in Ψ)

```
nil
|
a1
|
a2
|
a3
|
a4
|
a5
```

Lower bound, MFI/MII

- To describe the solution, we need the MFIs **or** the MIIs.
- However, we need to query **both** the MFIs **and** the MIIs to identify the result uniquely: $\Omega(|\text{MFI}| + |\text{MII}|)$ queries.
- We can have $|\text{MFI}| = \Omega(2^{|\text{MII}|})$ and vice-versa.
- This bound is **not tight** (e.g., chain).

