Declarative Sequential Pattern Mining in ASP

Thomas Guyet

▶ **To cite this version:**

Thomas Guyet. Declarative Sequential Pattern Mining in ASP. Conference on Inductive Logic Programming (ILP), Nov 2023, Bari, Italy. hal-04295035

HAL Id: hal-04295035
https://inria.hal.science/hal-04295035

Submitted on 20 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Declarative Sequential Pattern Mining in ASP

Enhance sequential patterns mining with background knowledge

Thomas Guyet

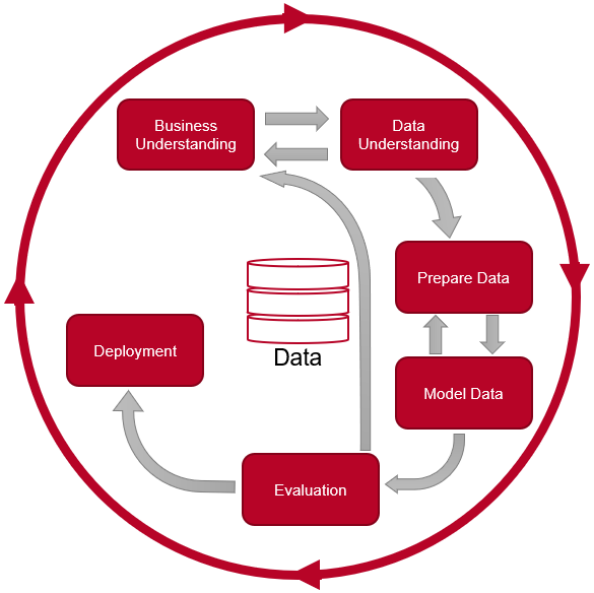
AIstroSight – Inria – Lyon Center

ILP@IJCLR 2023

Outline

General context of knowledge discovery

- Ultimate objective of data science is to discover new knowledge from data
  - Knowledge discovery is an **incremental process**
  - Combine both inductive and deductive reasoning
- Role of data science tools: **empower users**
  - User-centered/interactive process
    - Users create new pieces of knowledge
    - Tools provide him/her more capabilities: information access, integration, etc.

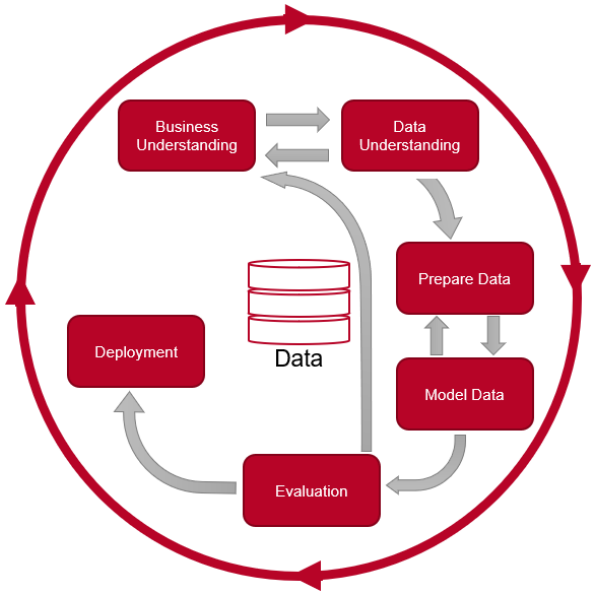


CRISP-DM cycle

https://michael-fuchs-python.netlify.app

General context of knowledge discovery

- Ultimate objective of data science is to discover new knowledge from data
  - Knowledge discovery is an **incremental process**
  - Combine both inductive and deductive reasoning
- Role of data science tools: **empower users**
  - User-centered/interactive process
    - Users create new pieces of knowledge
    - Tools provide him/her more capabilities: information access, integration, etc.



CRISP-DM cycle

https://michael-fuchs-python.netlify.app

- Many different users/questions/data, along with unexpected problems to overcome ⇒ necessitate **versatile tools**
- Moving toward intelligent assistant: incorporating more expert **reasoning** mechanisms into the tools

Pharmaco-epidemiology as a use-case

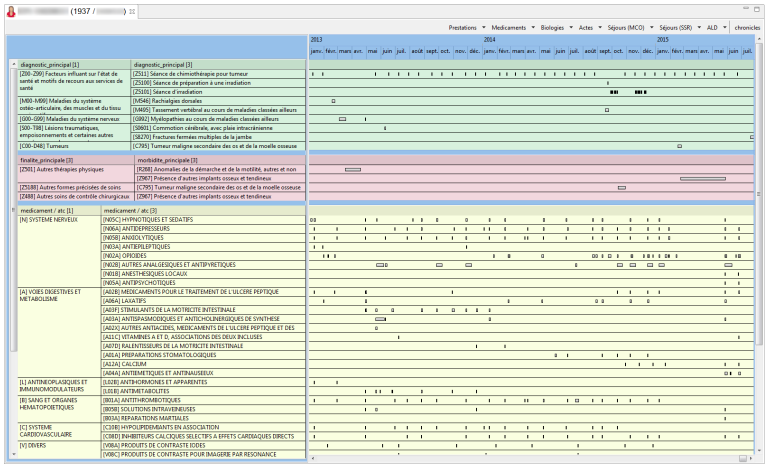
Objectives of Pharmaco-epidemiology

Discover correlations in real-life between health product consumption and health events

Use of Medico-administrative database

- Database of patients designed for administrative purposes
- Secondary usage for medical research (readily available)
- Longitudinal data

$\Rightarrow$ each patient is a **sequence of cares and medical events**



Sequential pattern mining applied to pharmaco-epidemiology

- A **sequential pattern** is an interesting piece of information about **co-occurrences of medical events**

$\rightarrow$ Frequent sequential patterns / discriminant sequential patterns

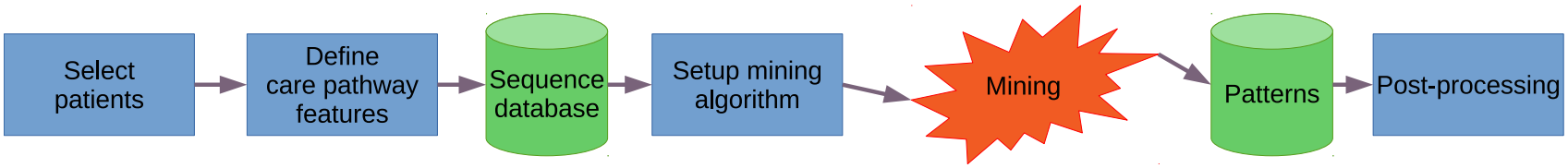
Pharmaco-epidemiology as a use-case

Pharmaco-epidemiology with sequential pattern mining

1. **processing steps prior to mining algorithm**: transforming a relational database into a set of sequences
   - select patients: define the *transactions*
   - select features (selection/proxy): defines the *items*
   - defining/setup the mining task (frequent/rare/discriminant?)
2. **mining the dataset of sequences**
3. **processing steps posterior to mining algorithm**: filtering outputs
   - selection/ranking of "best" patterns
   - qualify patterns

In pharmaco-epidemiological studies, the most challenging aspect is defining the pre-/post-processing of data analytics

Pharmaco-epidemiology with sequential pattern mining



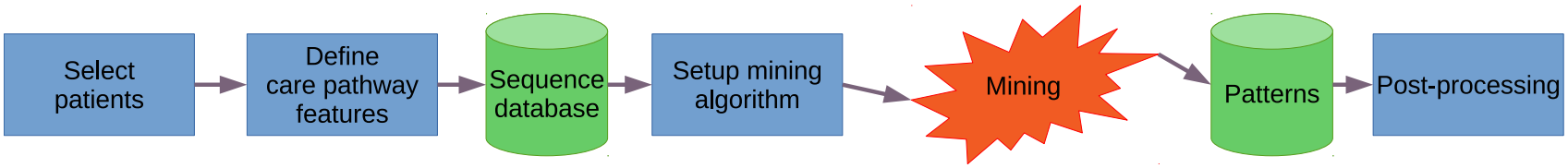
1. processing steps prior to the mining algorithms
2. mining the dataset of sequences
3. processing steps posterior to mining algorithms

In **pharmaco-epidemiological studies**, the most challenging aspect is defining the pre-/post-processing of pattern mining

It requires versatile tools:

- to ease the trials/errors approach
- to manage the complexity of medico-administrative databases (define complex proxies, *ontological reasoning*)
- to handle various analysis tasks

Pharmaco-epidemiology with sequential pattern mining



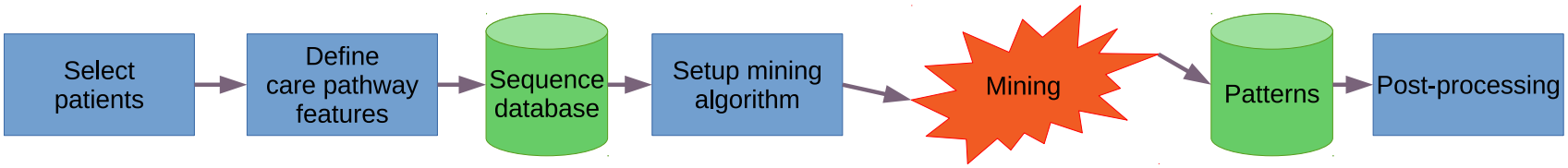
1. processing steps prior to the mining algorithms
2. mining the dataset of sequences
3. processing steps posterior to mining algorithms

In **pharmaco-epidemiological studies**, the most challenging aspect is defining the pre-/post-processing of pattern mining

It requires versatile tools: $\Rightarrow$ roots of inductive logic programming

- to ease the trials/errors approach
- to manage the complexity of medico-administrative databases (define complex proxies, *ontological reasoning*)
- to handle various analysis tasks

Pharmaco-epidemiology with sequential pattern mining



1. processing steps prior to the mining algorithms
2. mining the dataset of sequences
3. processing steps posterior to mining algorithms

In **pharmaco-epidemiological studies**, the most challenging aspect is defining the pre-/post-processing of pattern mining

It requires versatile tools: $\Rightarrow$ declarative programming approach

- to ease the trials/errors approach
- to manage the complexity of medico-administrative databases (define complex proxies, *ontological reasoning*)
- to handle various analysis tasks

Declarative pattern mining approach

Representing the pharmaco-epidemiology tool-chain in ASP

$$\langle \mathcal{D}, \mathfrak{S}, \mathcal{K}, \mathfrak{M}, \mathcal{C} \rangle$$

- $\mathcal{D}$ (facts): raw-administrative data base encoding
- $\mathcal{K}$ (facts): knowledge-base (ontology, guidelines, etc)
- $\mathfrak{S}$: rules for prior patient selection and feature selection
- $\mathfrak{M}$: efficient ASP encodings of sequence mining
- $\mathcal{C}$: pattern selection constraints

Answer Set Programming (ASP) sequence mining framework

- a unified formalism to represent the entire processing chain (*in few lines*)
- a versatile encoding that ease the trials and error approach
- a sufficiently efficient approach to experiment on mid-size datasets

Outline

Sequential pattern mining

- A **sequence *s*** is an ordered set of events (or **itemsets**)

SID	sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle(ab)c(bc)(ae)(ad)\rangle$
30	$\langle(ef)(ab)(df)cbc\rangle$
40	$\langle eg(af)cbc(de)\rangle$
50	$\langle aa(ad)b\rangle$

Sequential pattern mining

- A **sequence** s is an ordered set of events (or **itemsets**)
- A **sequential pattern** is a subsequence
  - example: pattern $p = \langle a(bc)d \rangle$
  - containment relation: $p \preceq s$
    - → inclusion of itemsets
    - → gaps are allowed
  - embedding: is a mapping of a pattern to a sequence ($\langle 1, 2, 4 \rangle$, $\langle 1, 3, 5 \rangle$)

SID	sequence
10	$\langle a(abc)(ac)d(cf) \rangle$
20	$\langle (ab)c(bc)(ae)(ad) \rangle$
30	$\langle (ef)(ab)(df)cbc \rangle$
40	$\langle eg(af)cbc(de) \rangle$
50	$\langle aa(ad)b \rangle$

Sequential pattern mining

- Let $\mathcal{D}$ be a set of sequences, the **support** of pattern $\boldsymbol{p}$ in $\mathcal{D}$ is the number of sequences in $\mathcal{D}$ that contain $\boldsymbol{p}$:

$$supp(\boldsymbol{p}) = |\{\boldsymbol{s} \in \mathcal{D} | \boldsymbol{p} \preceq \boldsymbol{s}\}|$$

- **Frequent pattern mining**: Given a support threshold σ, find the complete set of sequential patterns with support above σ

SID	sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle(ab)c(bc)(ae)(ad)\rangle$
30	$\langle(ef)(ab)(df)cbc\rangle$
40	$\langle eg(af)cbc(de)\rangle$

Some frequent patterns:

- $\langle abd\rangle$ (but not $\langle a(bc)d\rangle$)
- $\langle(ab)c\rangle$

Sequential pattern mining

- Let $\mathcal{D}$ be a set of sequences, the **support** of pattern $\boldsymbol{p}$ in $\mathcal{D}$ is the number of sequences in $\mathcal{D}$ that contain $\boldsymbol{p}$:

$$supp(\boldsymbol{p}) = |\{\boldsymbol{s} \in \mathcal{D} | \boldsymbol{p} \preceq \boldsymbol{s}\}|$$

- **Frequent pattern mining**: Given a support threshold σ, find the complete set of sequential patterns with support above σ

SID	sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ab)c(bc)(ae)(ad)\rangle$
30	$\langle (ef)(ab)(df)cbc\rangle$
40	$\langle eg(af)cbc(de)\rangle$

Some frequent patterns:

- $\langle abd\rangle$ (but not $\langle a(bc)d\rangle$)
- $\langle (ab)c\rangle$

↪ Algorithms are based on an **anti-monotonicity property**

$$\boldsymbol{p} \preceq \boldsymbol{p}' \implies supp(\boldsymbol{p}') \leq supp(\boldsymbol{p})$$

Integrating knowledge into pattern mining

Too many "useless patterns"

- pattern mining algorithms generate a **deluge of patterns** ...
- it is not reasonable to present this deluge of patterns to a data analyst
- a challenge of pattern mining: extract **fewer** but **meaningful patterns**.

Constrained(sequential) pattern mining

- A user defines constraints to specify the expected patterns based:
  - on previous results
  - on their expert knowledge
- Objective: refine the set of extracted patterns by adding constraints

A typology of constraints [NG15]

- **Pattern constraints**: expressed on an individual pattern $\boldsymbol{p}$
  - *examples:* exclude items, super-pattern, cost
  - $\rightarrow$ *extract only frequent patterns that do not contain the b item*
- **Embedding constraints**: specification of the embedding relation $\preceq$
  - *examples:* gap constraint, span constraint, number of embeddings
  - $\rightarrow$ $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$ *iff* $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$ *with at most two items in-between successive embedding positions* ($\langle(a)(bc)(ae)(e)(ad)\rangle$)
- **Dataset constraints**: expressed on the set of sequences matching a pattern $\{\boldsymbol{s} \mid \boldsymbol{s} \preceq \boldsymbol{p}\}$
  - *examples:* frequency constraint (frequent, rare), discriminant, contrastive
- **Pattern set constraints**: expressed on a set of patterns
  - *examples:* condensed representations (maximal, closed)
  - $\rightarrow$ *frequent pattern* $\boldsymbol{p}$ *without larger frequent pattern* $\boldsymbol{q}$, $\boldsymbol{p} \preceq \boldsymbol{q}$

Integrating knowledge to data mining process

- Difficulty: how to manage a wide range of complex constraints
  - $\rightarrow$ classically: one constraint = one algorithm!!

Main research lines to integrate constraints in pattern mining tasks

- Algorithms library [FVGG+14]
- Inductive databases [IM96]
- Pattern mining query language [De 15, BJ05, BL07, VCQ07, NG15]
- Declarative pattern mining
  - with SAT solver [CJSS12]
  - with CP solver [GDT+13, NG15, UBC+16]
  - with ASP solvers [Jär11, LS23, CT21, PSM19, KPA23]

Basic ASP encoding

Database encoding

- seq(***t***,*x*,*e*): sequence ***t***, at position *x* holds the item *e*

```
seq(0,1,20). seq(0,2,19). seq(0,3,2).  seq(0,4,18).
seq(1,1,5).  seq(1,2,6).  seq(1,3,6).  seq(1,4,14).
seq(2,1,13). seq(2,2,12). seq(2,3,9).
seq(3,1,15). seq(3,2,7).  seq(3,3,17). seq(3,4,17).
seq(4,1,11). seq(4,2,2).
seq(5,1,14). seq(5,2,8).  seq(5,3,13). seq(5,4,1).
```

Program constants

- #const max=6: maximal length of sequence (finite domain)
- #const k=10: support threshold

Basic ASP encoding of frequent sequential pattern mining

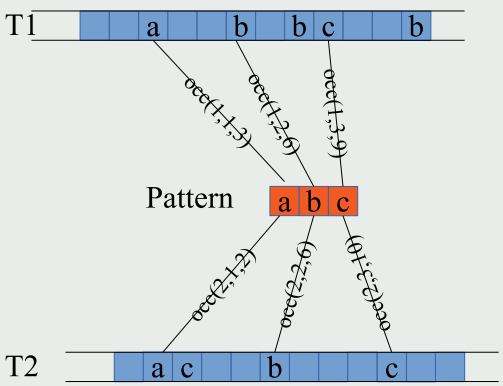
Atom	Meaning
item(e)	item e belongs to some $\boldsymbol{t} \in \mathcal{D}$
slot (x)	$1 \leq x \leq$ m refers to the position x of an item s_x in $\boldsymbol{p}$
pat(x,e)	$p_x = e$ is the item at position x in $\boldsymbol{p}$
cover($\boldsymbol{t}$)	$\langle p_1 \ \ldots \ p_m \rangle \preceq \langle t_1 \ \ldots \ t_n \rangle$, that is, $\boldsymbol{p} \preceq \boldsymbol{t}$
occ($\boldsymbol{t}$,p,x)	$\exists (e_i)_{i \in [p-1]}$ such that $p_i \subseteq t_{e_i}$ for all $i \in [p-1]$ and $p_x \subseteq t_p$.

```
1  item(I) :- seq(T,P,I).

3  slot(1).
4  { slot(X+1) } :- slot(X); X<max.
5  1{ pat(X,E) : item(E) } :-
     slot(X).

7  patlen(L) :- pat(L,_);
8               not pat(L+1,E):item(E).

9  occ(T,P,1) :- seq(T,P,E):pat(1,E); seq(T,P,_).
10 occ(T,P,X+1) :- occ(T,Q,X); seq(T,P,_); Q<P;
11                 seq(T,P,E):pat(X+1,E);
12                 patlen(L); X<L.

14 cover(T) :- patlen(L); occ(T,_,L).

16 :- not k { cover(T) }.
```

Original encoding: skip-gaps strategy

occ(t,p,x): $\exists (e_i)_{i\in[p-1]}$ such that $p_i \subseteq t_{e_i}$ for all $i \in [p-1]$ and $p_x \subseteq t_p$.

Example: description of transaction occurrences




```
pat(a). pat(b). pat(c).
occ(1,1,3). occ(1,2,6). occ(1,3,9).
occ(2,1,2). occ(2,2,6). occ(2,3,10).
```

⏩ fillgaps alternative

Alternative encodings I

Several alternative encodings were implemented and evaluated [GMQS16]

- Alternative strategies to evaluate pattern covering
  - skip-gaps vs fill-gaps embedding evaluations
- Efficient additional rules
  - Early filtering out infrequent items
  - Using projected databases principle: extend patterns only with potentially frequent items (fill-gap basis)

```
item(1,I)      :- item(I),
                  #count{ T: seq(T,_,I) } >= th.
item(Ip+1,I)   :- item(Ip,I),
                  #count{ T: seq(T,Is,I), occ(T,Ip,Is) } >= th.
1 { pat(Ip,I) : item(Ip,I) } 1 :- slot(Ip)
```

+ Use some (not intuitive) tricks to improve practical efficiency

Alternative encodings II

Additional constraints on the tasks (some examples)

- constrained sequential patterns
  - pattern constraints (efficient and easy to implement constraints)
    `:- pat(X,I), forbidden_item(I).`
  - more practical examples on the use-case

Alternative tasks encodings

- closed and maximal patterns [GGQ+16, GMQS16]
- rare sequential patterns [SGN17]
- negative sequential patterns [BG20]
- contrastive patterns [LS23]
- case-crossover patterns

$\Rightarrow$ **Most of the constraints only require the addition of rules**

- easy to modify a pattern mining task by (des)activate rules

Are ASP pattern mining ready for practical use?

Many experiments conducted on real and synthetic datasets [GMQS16]

Some evaluation of encodings

- Two different encodings for occurrences search
  - `sg`: basic encoding
  - `fg`: evaluating occurrences with the *fillgap* strategy
- Constraints: `None`, `Maximal` or `Closed`
- Solving heuristic: `normal`, subset minimal (`dom`) possibly incomplete
- CPSM: Constraint Programming for Sequence Mining [NG15]

Evaluation criteria

- computing efficiency: time to extract all the patterns
- memory usage: number of grounded atoms / maximum allocated memory

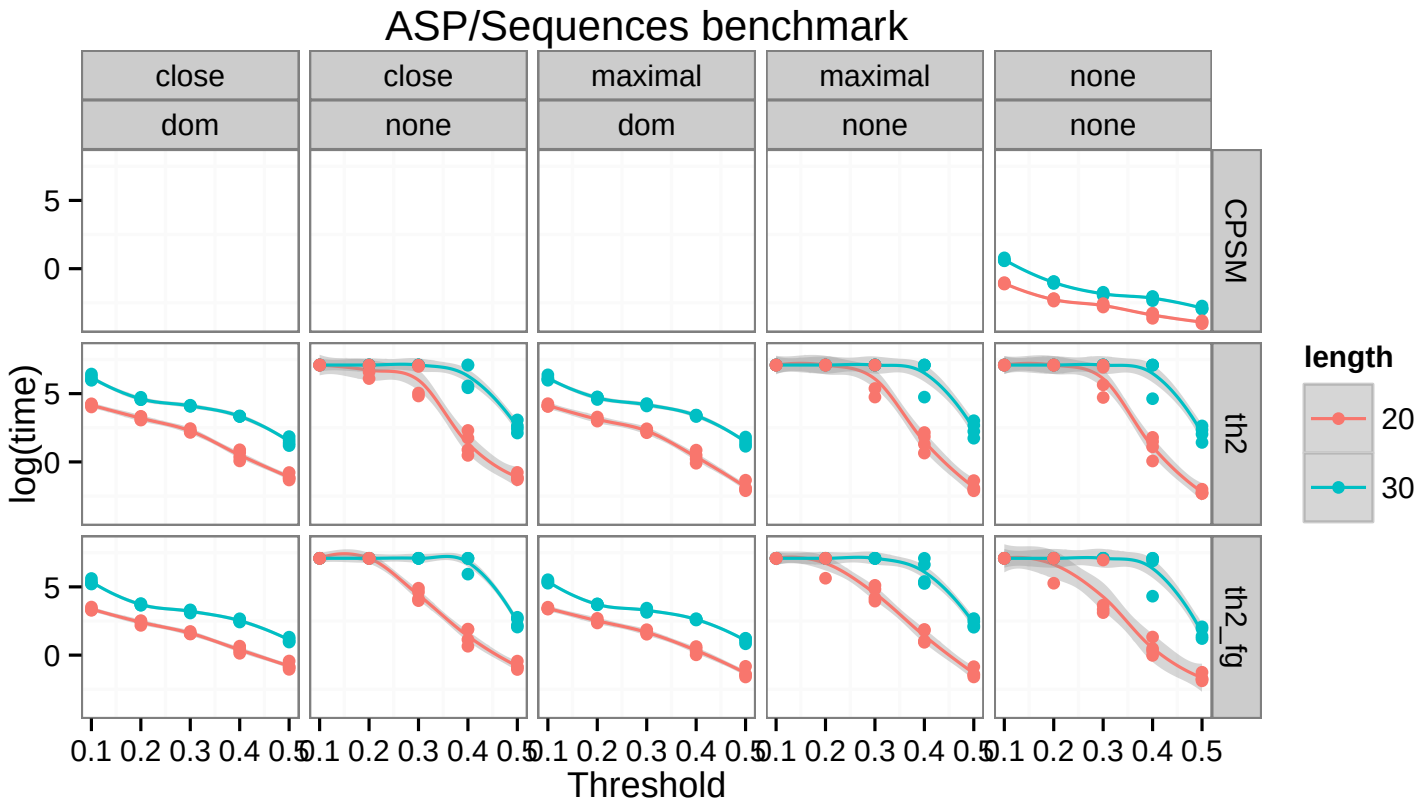
Simulated dataset

- Dataset generation
  - database of 1000 sequences, 100 different items
  - mean length of sequences: 30 items
  - 20 simulated patterns of mean length 5 and with a frequency of 10%
  - probability of occurrence of an item: a normal distribution ($\mu = .5$, $\sigma = 0.05$)
- Solver setting
  - clingo 4.5
  - 20Go RAM, 4 threads (server)
  - timeout: 20 minutes

⏩ More details

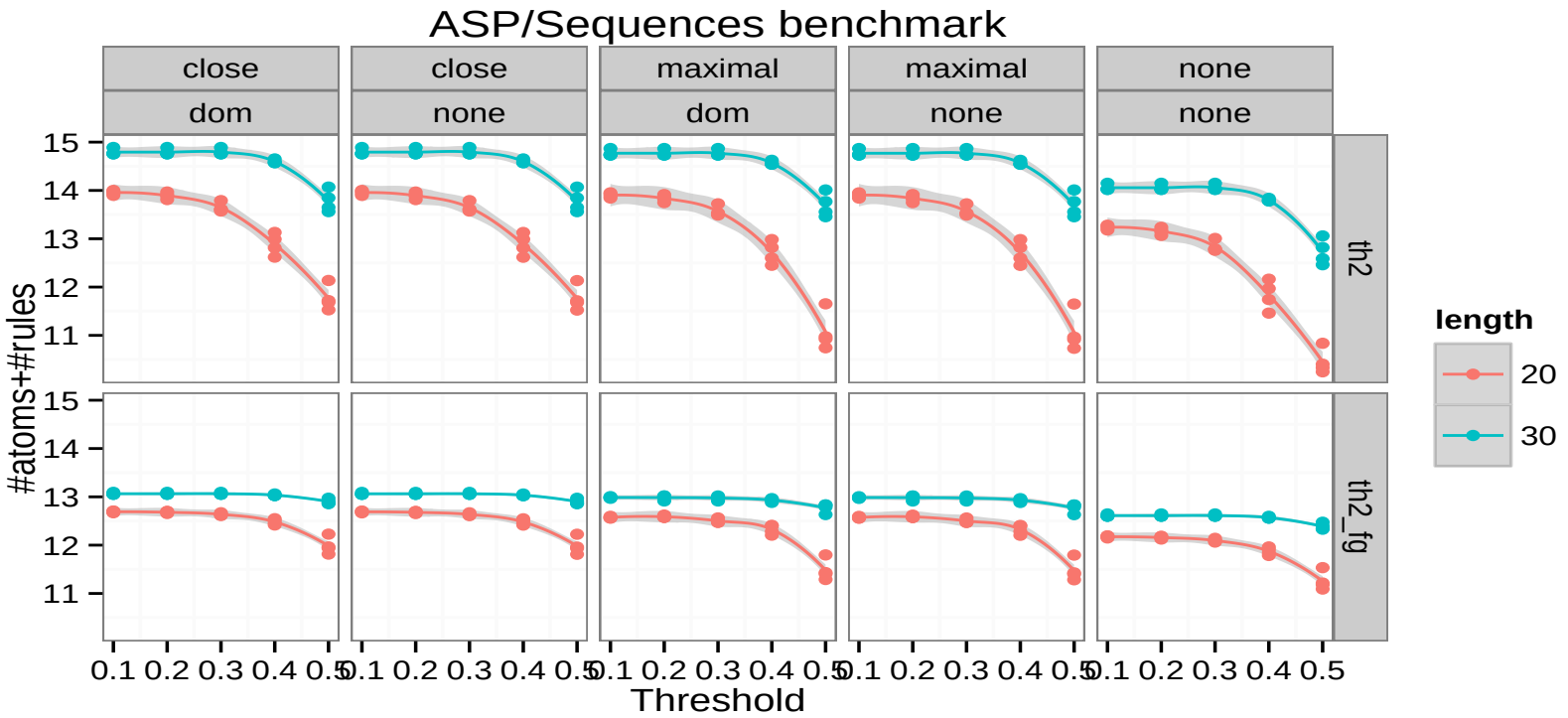
⏩ Results on real-world datasets

Results - Computing time



- fill-gaps is slightly more efficient (especially for large memory requirement problems)
- CPSM is 1 to 2 order of magnitude faster than ASP

Results - Memory usage



- fill-gaps encoding reduces significantly the required memory
- **length of the sequences is a very sensitive** dataset characteristic
- very low threshold will not increase the memory requirements
- maximal and closed encoding require more memory

Partial conclusions

- ASP enables to encode sequential pattern mining tasks in an "intuitive" way
- Different encodings have been proposed
  - easy implementation/(des)activation of a wide range of pattern mining constraints

  $\Rightarrow$ versatility and readability of encodings
- Evaluation of the efficiency, ASP encoding of FSM is
  - *not* competitive with state of the art algorithms
  - comparable with "fairly-versatile" other declarative frameworks (SAT/CP)
- Some limitations

  $\Rightarrow$ memory requirement is the hardest constraint

  $\Rightarrow$ the most efficient encodings are not the most "intuitive"

Outline

Declarative pattern mining approach (*reminder*)

Representing the pharmaco-epidemiology tool-chain

$$\langle \mathcal{D}, \mathfrak{S}, \mathcal{K}, \mathfrak{M}, \mathcal{C} \rangle$$

- $\mathcal{D}$ (facts): raw-administrative data base encoding
- $\mathcal{K}$ (facts): knowledge-base (ontology, guidelines, etc)
- $\mathfrak{S}$: rules for prior patient selection
- $\mathfrak{M}$: efficient ASP encodings of sequence mining (first part of this talk)
- $\mathcal{C}$: pattern selection constraints

Case-study: GENEPI study [GHD17]

GENEPI: a pharmaco-epidemiology study

- Context: Epileptic patients having an antiepileptic treatment
- Objective: identify possible associations between hospitalization for seizure and antiepileptic drug (AED) change in the AED treatments
- GENEPI investigates changes between generic and brand-name AE drugs

French database of epileptic patients

- pre-selected patients with epileptic seizures
- 8,379 patients
- 20,686 seizure-related events
- 1,810,600 deliveries of 7,693 different drugs hospitalizations

Raw database of patients ($\mathcal{D}$) in FOL

Patient *raw* database

- Patient information (e.g. sex, age, chronic long-term illness)
- Care events (timestamped):
  - hospital stays (including diagnosis codes)
  - drugs deliveries
  - medical consultations (general practitioners, specialists)

Facts encoding:

```
patient(0).

sex(0,m).
birthYear(0,1960).

delivery(0,1,3585053,1).
delivery(0,1,3599730,1).
delivery(0,154,3599730,1).
delivery(0,346,3599730,1).
delivery(0,350,3599730,2).

disease(0,380,g409).
disease(0,380,k700).
```

Knowledge encoding:

```
%Details about drug CIP 3585053 (Tramadol)
cip_atc(3585053,r06ae09). % ATC class of Tramadol: R09AE09
grs(3585053,364).         % Speciality group identifier (GRS)
generic(3585053).         % This CIP is a generic drug
nbpills(3585053,28).      % 28 doses per blister
dosage(3585053,5).        % 5mg per dose

%ATC Taxonomy for R06AE09 ATC code
is_a(r06ae09, r06ae). is_a(r06ae, r06a). is_a(r06a, r06).
```

Data preparation $\mathfrak{S}$

Sequence of AED deliveries within the 3 months before the **first seizure**

```
firstseizure(P,T) :-disease(P,T,D), is_a(D,g40;g41),
                    #count{Tp: disease(P,Tp,Dp), is_a(Dp,g40;g41), Tp<T}=0.

seq(P,T,delivery(AED,1)):- delivery(P,T,CIP,Q), aed(CIP,AED), generic(CIP),
                           T<Ts, T>Ts-90, firstseizure(P,Ts).

seq(P,T,delivery(AED,0)):- delivery(P,T,CIP,Q), aed(CIP,AED), not generic(CIP),
                           T<Ts, T>Ts-90, firstseizure(P,Ts).
```

- *G*40 and *G*41 are disease codes for epileptic seizures (*CIM-10*)
- aed/2: classifies a drug code in one of the 5 ATC classes of AED
- dates are numbers of days (integers)

$\Rightarrow$ yield seq/3 atoms required by pattern mining encoding

$\rightarrow$ 1300 patterns ($f_{min} = 42$) $\rightarrow$ It's too much

32.8s

Pattern selection $\mathcal{C}$

Pattern constraints: two types of drugs

```
:- #count{X : pat(X,delivery(AED,0)) }=0. % has a generic drug
:- #count{X : pat(X,delivery(AED,1)) }=0. % has a brand-name drug
```

→ 151 models

29.8s

Embedding constraints: Max-gap constraint (at most 35 days)

```
#const maxgap=35.
:- occ(S,X,T), occ(S,X+1,Tp), Tp>T+maxgap, pat(X,C), pat(X+1,D).
```

→ 36 models (only 10 of length ≥ 3) → can be investigated by a clinician

89.2s

⇒ Much more possibilities to (easily) encode new pattern mining tasks

case-crossover

Partial conclusions

- ASP enables to encode sequential pattern mining tasks in an "intuitive" way
- **Proposition of an end-to-end data analysis toolchain in ASP**
  - ASP language: uniform formalism for data, background knowledge and processings
  - First-order language eases the representation of data and kwowledge (compare to other declarative frameworks)
  - Very powerful solution for **fast prototyping of a data science tool chain**
- Evaluation of the efficiency, ASP encoding of FSM is
  - *not* competitive with state-of-the-art algorithms
  - comparable with "fairly-versatile" other declarative frameworks
  - **improved by adding (even multiple/complex) tool chain constraints**
    - $\rightarrow$ constraints benefit to improve all toolchain steps
- Some limitations
  - $\Rightarrow$ memory requirement is the hardest constraint **even more than before**
  - $\Rightarrow$ the most efficient encodings are not the most "intuitive"

Outline

Diagnosis of massive memory consumption

Reminder about ASP program solving

- Two-stage solving procedure
  1. **Grounding**: the first order program is (smartly) "flatten" into a boolean problem
  2. **Solving**: solve the boolean problem to find stable answer sets
- Solver strategies
  - some solvers can make grounding on the fly (`asperix` [LBSG17])
  - `clingo` grounds the problem **in memory**

$\Rightarrow$ evaluation of pattern's embeddings is memory consuming

```
1  item(I) :- seq(T,P,I).

3  slot(1).
4  { slot(X+1) } :- slot(X); X<max.
5  1{ pat(X,E) : item(E) } :-
   slot(X).

7  patlen(L) :- pat(L,_);
8               not pat(L+1,E):item(E).

9  occ(T,P,1) :- seq(T,P,E):pat(1,E); seq(T,P,_).
10 occ(T,P,X+1) :- occ(T,Q,X); seq(T,P,_); Q<P;
11                 seq(T,P,E):pat(X+1,E);
12                 patlen(L); X<L.

14 cover(T) :- patlen(L); occ(T,_,L).

16 :- not k { cover(T) }.
```

Diagnosis of massive memory consumption

Reminder about solving ASP programs

- Two-stage solving
  1. **Grounding**: the first order program is "flatten" into a boolean problem
  2. **Solving**: solve the boolean problem
- Solver strategies
  - some solvers can make grounding on the fly (`asperix` [LBSG17])
  - `clingo` grounds the problem **in memory**

$\Rightarrow$ evaluation of pattern's embeddings is memory consuming

Toward a memory-light solution

- we mainly need to evaluate the sequence covering constraint: $\boldsymbol{p} \preceq \boldsymbol{s}$
- it does not require to expose the embedding details

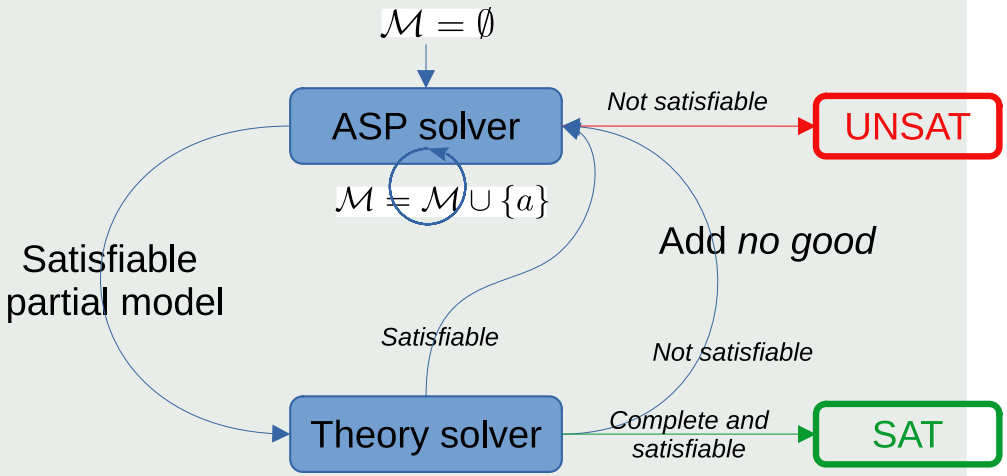
$\rightarrow$ can we evaluate sequence covering without computing the embeddings?

ASP modulo Theory (ASP-MT) with `clingo`

ASP modulo Theory (ASP-MT) in `clingo` [GKK+16]

- ASP programs and theories
  - ASP program P: defines atoms and logical constraints on atoms
  - Theory: additional constraints Δ_T on some program atoms
    - not expressed as ASP rules
    - **an external *entity* is in charge to assess whether constraints Δ_T are satisfied or not** $\rightarrow$ propagator
    - internal view of the program P

- ASP-MT in `clingo 5`
  - Theory specification language
  - Theory propagators
  - Extension of the Conflict-Driven Constraint Learning (CDCL) algorithm to ASP-MT



ASP-MT for pattern mining (*ongoing work*)

Program modification

- No more occ/3 atoms in the encoding
- cover/1 is a theory atom
  - in ASP program: it can be true or false
  - the theory propagator checks the constraint: $\text{cover}(s) \Leftrightarrow p \preceq s$

```
1  item(I) :- seq(_,_,I).

3  slot(0..minlen).
4  { slot(X+1) } :- slot(X), X<maxlen-1.
5  patlen(L+1) :- slot(L), not slot(L+1).

7  1 { pat(X,I) : item(I) } 1 :- slot(X).

9  % is the pattern covers the sequence S or not??
10 { cover(S) } :- seq(S,_,I), pat(_,I).

12 :- #count{ S : cover(S) } < th.
```

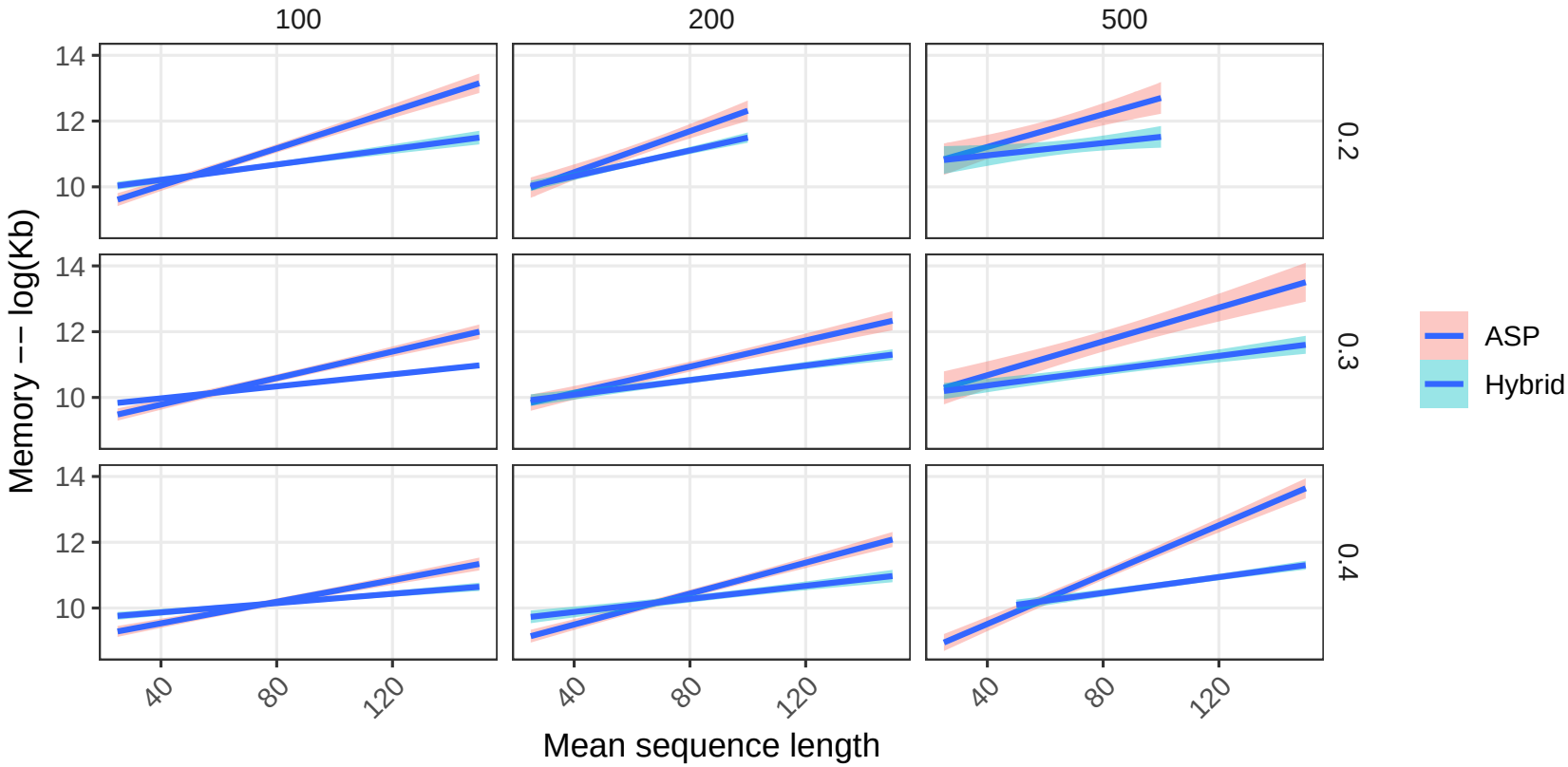
ASP-MT for pattern mining: sequence covering propagator

- Constraint to check: $\text{cover}(s) \Leftrightarrow p \preceq s$
  - $\text{cover}(s)$ and p are defined in the model
  - $p \preceq s$ is checked with a procedural algorithm (Python)
- Two types of conflict
  - $\text{cover}(s) \wedge p \not\preceq s$: faulty covered sequence in the model
  - $\neg\text{cover}(s) \wedge p \preceq s$: missing covered sequence in the model
- Role of the theory propagator
  - $\rightarrow$ **detect conflicts** with the theory constraint in a partial model
  - $\rightarrow$ **identify *no goods***: a small subset of atoms that "explains" a conflict
- The "smallest" the *no good*, the more informative $\Rightarrow$ detect conflicts from partial models

Early detection of conflicts in partial models

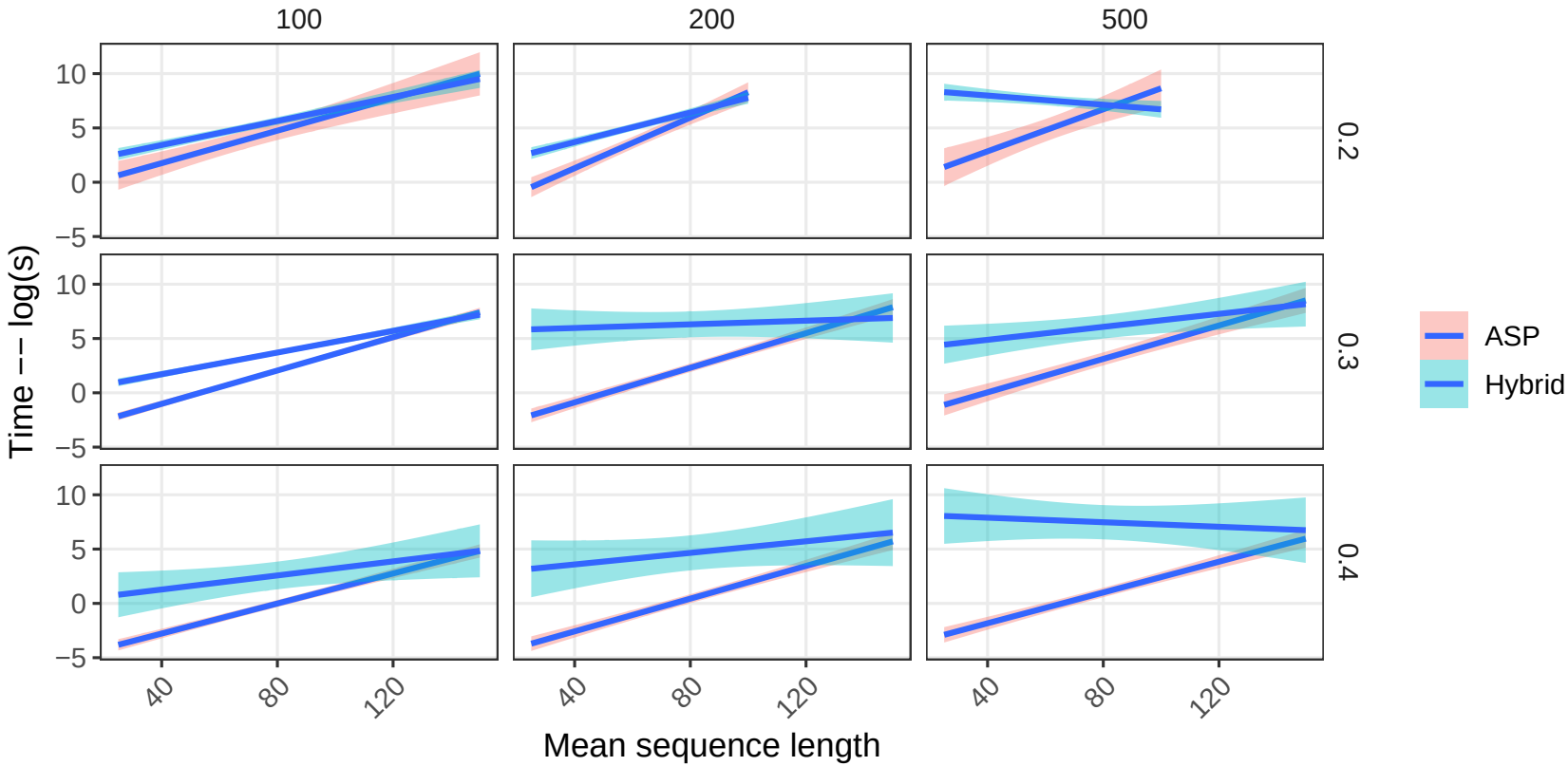
- Partial model: describe a pattern q s.t. $q \preceq p$
  - $\text{cover}(s) \wedge q \not\preceq s$, then $p \not\preceq s$: an early *no-good* is send to the solver
  - $\neg\text{cover}(s) \wedge q \preceq s$, ...: *can not but used as no good to discard* $p \preceq s$

Comparison between propagators and pure ASP



+ Propagator requires less memory, especially for long sequences
+ The dataset size has a limited impact on propagator memory consumption

Comparison between propagators and pure ASP



+ Propagator requires less memory, especially for long sequences
+ The dataset size has a limited impact on propagator memory consumption
- The propagator is less time-efficient than pure ASP (!)

Comments about the versatility of the proposed model

Different tradeoff between declarative solvers and theory solvers

- frequent pattern "propagator" [CJSS12]: the theory assesses an atom frequent($\boldsymbol{p}$)
- **sequence covering "propagator"**: the theory assesses an atom cover($\boldsymbol{s}$)
- propagator with embeddings exposition (in CP [NG15])

Our ASP-MT encoding enables to extend the program with:

- pattern constraints: *ASP rules*
- pattern set constraints: *ASP rules*
- datasets constraints: *ASP rules*
- embedding constraints ($\boldsymbol{p} \sqsubseteq \boldsymbol{s}$):
  - embeddings are not exposed in the ASP encoding
  - extend propagator options + theory language (hardly coded, less versatile)

$\Rightarrow$ **it is compatible with (most of) the end-to-end toolchains**

Outline

Conclusions

- ASP enables to encode sequential pattern mining tasks in an "intuitive" way
  - **Expressivity**: various sequential pattern mining tasks have been encoded
    - classical tasks: closed, maximal, emergent, sky-patterns, etc.
    - pattern constraints encodings
    - possible design of new mining tasks case-crossover
  - `clingo` solver: operational, efficient $\Rightarrow$ realistic on (mid-size) real dataset

Conclusions

- ASP enables to encode sequential pattern mining tasks in an "intuitive" way
  - **Expressivity**: various sequential pattern mining tasks have been encoded
  - `clingo` solver: operational, efficient $\Rightarrow$ realistic on (mid-size) real dataset
- Proposition of an end-to-end data analysis toolchain in ASP
  - $\langle \mathcal{D}, \mathfrak{S}, \mathcal{K}, \mathfrak{M}, \mathcal{C} \rangle$
  - Illustrates the real interest of declarative sequential pattern mining
  - ... **versatility of declarative SPM for fast task prototyping**
  - Possibility to create new mining tasks: case-crossover patterns

Conclusions

- ASP enables to encode sequential pattern mining tasks in an "intuitive" way
  - **Expressivity**: various sequential pattern mining tasks have been encoded
  - `clingo` solver: operational, efficient $\Rightarrow$ realistic on (mid-size) real dataset
- Proposition of an end-to-end data analysis toolchain in ASP
  - ... **versatility of declarative SPM for fast task prototyping**
- **Investigation of ASP-MT** to address the problem of heavy memory consumption
  - "sequence covering" propagator
  - Firsts encouraging results
  - Possibility to create new tasks: case-crossover patterns

Perspectives

Main perspective: Epistemic Measure of Interestingness (*ongoing work*)

Toward deeper integration of reasoning within the mining: a opportunity to develop **epistemic measure of interestingness** to tackle the major issue of pattern mining: the deluge of patterns.

Intuition: extract "interesting" patterns

- "interesting" is often a statistical measure (frequent, rare, emergent)
- Tilj De Bie extended the notion of interestingness to notions such as "novelty": again, it is based on statistical measure (a bayesian models)

→ my proposal is to define the *notion of "interestingness" based on prior formalized knowledge (epistemic models)*

ASP is suitable to implement reasoning techniques based on knowledge

→ could make the difference with other declarative frameworks

→ combining both (complex) reasoning and mining

Thanks for your attention!

Questions?

Thanks to collaborators: Potsdam team (T. Schaub, M. Gebser, R. Kaminsky), Y. Moinard, Ph. Besnard, J. Nicolas, R. Quiniou, A. Samet, Epidemiologists team (A. Happe, E. Oger).

References I


Philippe Besnard and Thomas Guyet, *Declarative mining of negative sequential patterns*, 1st Declarative Problem Solving Workshop (DPSW 2020)@ ECAI 2020, 2020, pp. 1–8.


Jean-Francois Boulicaut and Baptiste Jeudy, *Constraint-based data mining*, Data Mining and Knowledge Discovery Handbook (Oded Maimon and Lior Rokach, eds.), Springer US, 2005, pp. 399–416.


Francesco Bonchi and Claudio Lucchese, *Extending the state-of-the-art of constraint-based pattern discovery*, Data Knowl. Eng. **60** (2007), no. 2, 377–399.


Emmanuel Coquery, Said Jabbour, Lakhdar Saïs, and Yakoub Salhi, *A SAT-Based approach for discovering frequent, closed and maximal patterns in a sequence*, 20th European Conference on Artificial Intelligence (ECAI'12), 2012, pp. 258–263.


Francesco Cauteruccio and Giorgio Terracina, *An answer set programming based framework for high-utility pattern mining extended with facets and advanced utility functions*, Rules and Reasoning: 5th International Joint Conference, RuleML+ RR 2021, Leuven, Belgium, September 13–15, 2021, Proceedings 5, Springer, 2021, pp. 126–141.


Luc De Raedt, *Languages for learning and mining*, Proceedings of the Conference on Artificial Intelligence (AAAI), 2015, pp. 4107–4111.


Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Cheng-Wei Wu, Vincent S Tseng, et al., *Spmf: a java open-source pattern mining library.*, J. Mach. Learn. Res. **15** (2014), no. 1, 3389–3393.


Tias Guns, Anton Dries, Guido Tack, Siegfried Nijssen, and Luc De Raedt, *MiningZinc: A modeling language for constraint-based mining*, Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 2013, pp. 1365–1372.


Martin Gebser, Thomas Guyet, René Quiniou, Javier Romero, and Torsten Schaub, *Knowledge-based Sequence Mining with ASP*, IJCAI 2016- 25th International joint conference on artificial intelligence (New-york, United States), AAAI, 2016, p. 8.


Thomas Guyet, André Happe, and Yann Dauxais, *Declarative Sequential Pattern Mining of Care Pathways*, Conference on Artificial Intelligence in Medicine in Europe, 16th Conference on Artificial Intelligence in Medicine, vol. 24, 2017, pp. 1161 – 266.

References II


Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko, *Theory solving made easy with clingo 5*, Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.


Thomas Guyet, Yves Moinard, René Quiniou, and Torsten Schaub, *Efficiency analysis of ASP encodings for sequential pattern mining tasks*, Advances in Knowledge Discovery and Management - Volume 7 [Best of EGC 2016,Reims, France] (Bruno Pinaud, Fabrice Guillet, Bruno Crémilleux, and Cyril de Runz, eds.), Studies in Computational Intelligence, vol. 732, 2016, pp. 41–81.


Tomasz Imielinski and Heikki Mannila, *A database perspective on knowledge discovery*, Communications of the ACM **39** (1996), no. 11, 58–64.


Matti Järvisalo, *Itemset mining as a challenge application for answer set enumeration*, Logic Programming and Nonmonotonic Reasoning, Springer, 2011, pp. 304–310.


Nikos Katzouris, Georgios Paliouras, and Alexander Artikis, *Online learning probabilistic event calculus theories in answer set programming*, Theory and Practice of Logic Programming **23** (2023), no. 2, 362–386.


Claire Lefèvre, Christopher Béatrix, Igor Stéphan, and Laurent Garcia, *Asperix, a first-order forward chaining approach for answer set computing*, Theory and Practice of Logic Programming **17** (2017), no. 3, 266–310.


Francesca Alessandra Lisi and Gioacchino Sterlicchio, *Mining contrast sequential patterns with asp*, International Conference of the Italian Association for Artificial Intelligence, Springer, 2023, pp. 44–57.


Benjamin Negrevergne and Tias Guns, *Constraint-based sequence mining using constraint programming*, Proceedings of International Conference on Integration of AI and OR Techniques in Constraint Programming, CPAIOR, 2015, pp. 288–305.


Sergey Paramonov, Daria Stepanova, and Pauli Miettinen, *Hybrid asp-based approach to pattern mining*, Theory and Practice of Logic Programming **19** (2019), no. 4, 505–535.


Ahmed Samet, Thomas Guyet, and Benjamin Negrevergne, *Mining rare sequential patterns with ASP*, ILP 2017 - 27th International Conference on Inductive Logic Programming (Orléans, France), September 2017.

References III

Willy Ugarte, Patrice Boizumault, Bruno Crémilleux, Alban Lepailleur, Samir Loudni, Marc Plantevit, Chedy Raïssi, and Arnaud Soulet, *Skypattern mining: From pattern condensed representations to dynamic constraint satisfaction problems*, Artificial Intelligence (2016), to appear.

Alexandre Vautier, Marie-Odile Cordier, and Rene Quiniou, *Towards data mining without information on knowledge structure*, Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases, 2007, pp. 300–311.

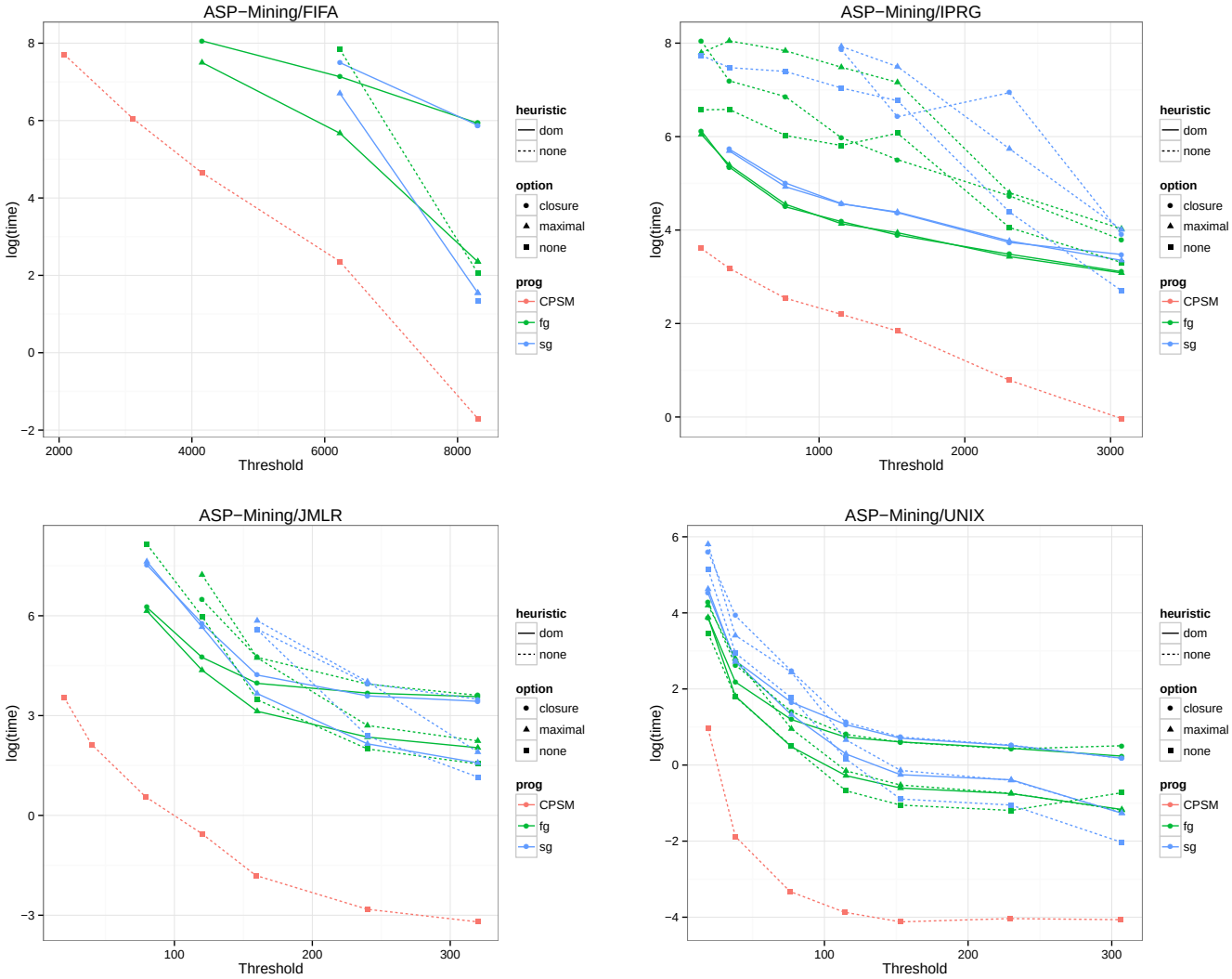
Experiment details

Simulated datasets

Return

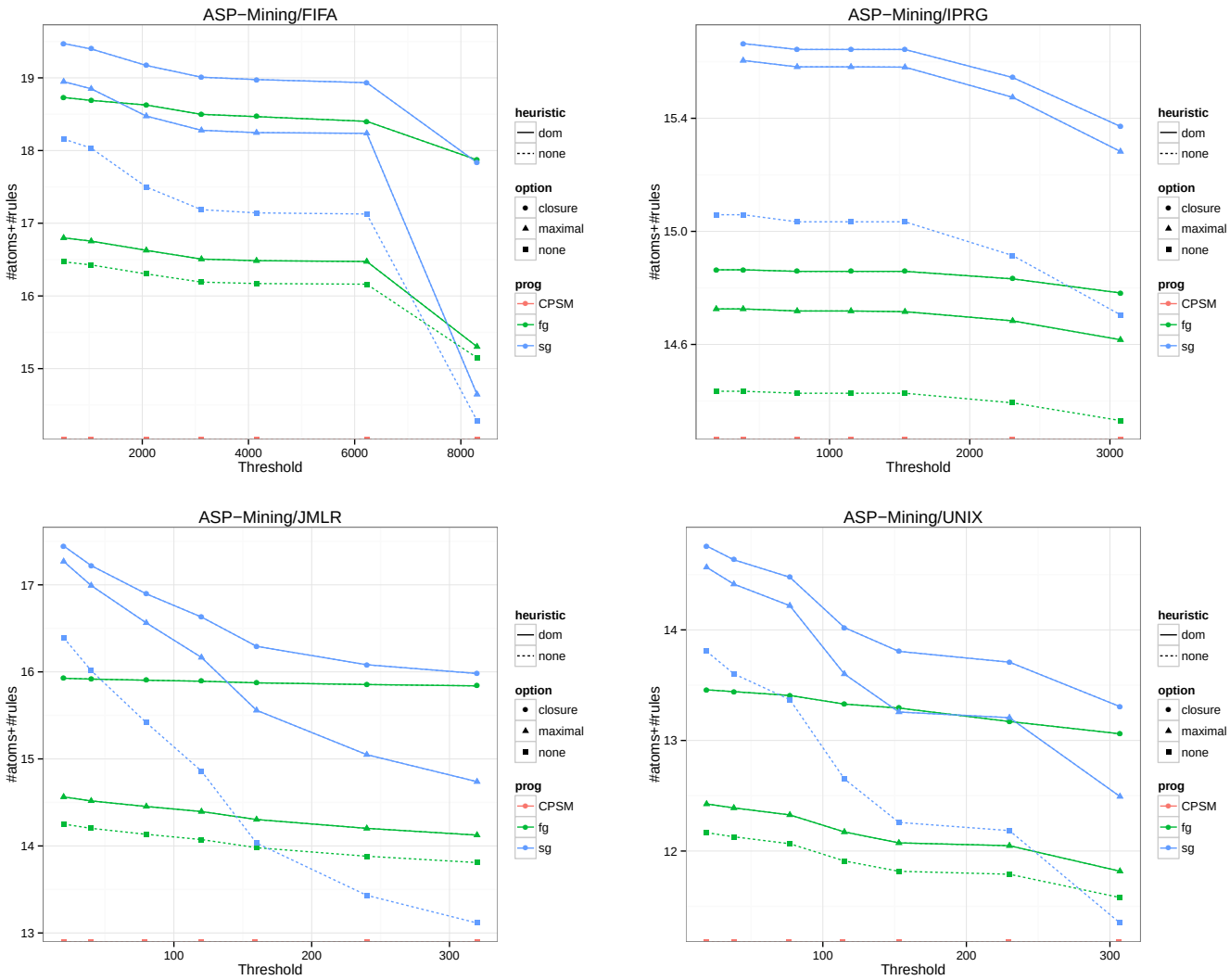
- "retro-engineering" simulation process
  1. a set of random patterns are generated
  2. a random number of occurrences of a pattern are assigned to the sequences of the database
  3. each sequence the database are generated
     - if not any occurrence of a pattern, then a sequence of random itemsets is generated,
     - otherwise, the items of the sequence are sequentially generated (for each successive position). The item can be an item of an occurrence of a pattern or an random item.
- main parameters of the simulated dataset
  - D: the number of sequences in the database
  - l: the mean length of the sequence (normal law)
  - n: the number of patterns
  - lp: the mean length of a pattern
  - th: the minimum number of occurrences of each pattern
  - k: the size of the vocabulary (number of different items in the database). The distribution of the occurrence probability of the items are uniform (with the same probability) or normal.

Realworld dataset experiments – Time



Return

Realworld dataset experiments – Memory consumption

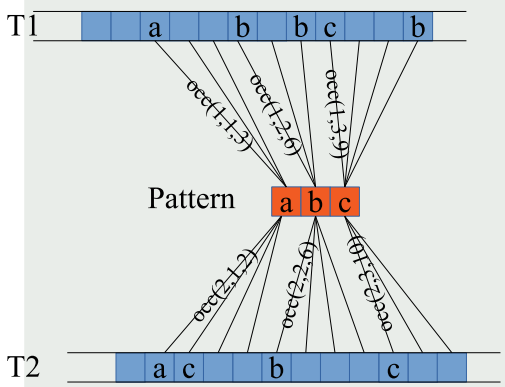


Return

Alternative encoding: fill-gaps strategy

occ(t,p,x): $\exists (e_i)_{i\in[p]}$ such that $p_i \subseteq t_{e_i}$ for all $i \in [p]$ ~~and $p_x \subseteq t_p$~~.

Description of the occurrences: second solution, the **fill-gaps** solution



skip-gap

```
%pattern embeddings
occ(T,1,Is)  :- seq(T,Is,I), pat(1,I).
occ(T,Ip,Is) :- occ(T, Ip-1, Is-1), seq(T,Is,I),
pat(L,I).
occ(T,Ip,Is) :- occ(T, Ip,   Is-1), seq(T,Is,_).

%frequency constraint
seqlen(T,L) :- seq(T,L,_), not seq(T,L+1,_).
support(T) :- occF(T, L, LS), patlen(L), seqlen(T,LS).
:- { support(T) } < th.

pat(a). pat(b). pat(c).
occ(1,1,3). occ(1,1,4). occ(1,1,5). occ(1,2,6).
occ(1,2,7). occ(1,2,8). occ(1,3,9).
occ(2,1,2). occ(2,1,3). occ(2,1,4). occ(2,1,5).
occ(2,2,6). occ(2,2,7). occ(2,2,8).
occ(2,2,9). occ(2,3,10).
```

Encoding discriminant patterns

Case-crossover setting

- Case period: 3 months before the seizure
- Control period: from 6 to 3 months before the seizure

Emerging patterns encoding (growth rate g_{min})

```
nbpat(N) :-N=#count{P:disease(P,_,_)}.
% positive sequences
seq(P,T,delivery(AED,GRS,G)) :-delivery(P,T,I,Q), cip_code(I,AED,GRS,G),
                               firstseizure(P,ST), T<ST, T>ST-90.
% negative sequences
seq(P+N,T,delivery(AED,GRS,G)) :-delivery(P,T,I,Q), cip_code(I,AED,GRS,G),
                              firstseizure(P,ST), T<ST-90, T>ST-180, nbpat(N).


#const gmin=4. % give a threshold of 1+gmin/10
suppos(N) :-N={ support(T) : T<=NP }, nbpat(NP).
supneg(N) :-N={ support(T) : T>NP }, nbpat(NP).
:- suppos(P), supneg(N), 10*P<N*(10+gmin).
```

Encoding of a new mining task: case-crossover

Case-crossover setting

- Case period: 3 months before the seizure
- Control period: from 6 to 3 months before the seizure

Case-crossover frequent patterns encoding

```
nbpat(N) :-N=#count{P:disease(P,_,_)}.
% positive sequences
seq(P,T,delivery(AED,GRS,G)) :-delivery(P,T,I,Q), cip_code(I,AED,GRS,G),
                               firstseizure(P,ST), T<ST, T>ST-90.
% negative sequences
seq(P+N,T,delivery(AED,GRS,G)) :-delivery(P,T,I,Q), cip_code(I,AED,GRS,G),
                              firstseizure(P,ST), T<ST-90, T>ST-180, nbpat(N).

discr(T) :-support(T), not support(N+T), T<N, nbpat(N).
:- { discr(T) } < th.
```

return

Epistemic Measure of Interestingness: an example I

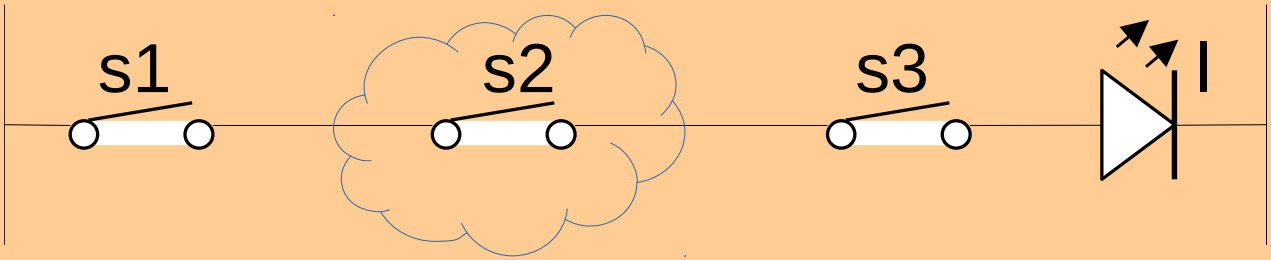
An example: temporal reasoning and pattern mining

- The context: understanding fault of industrial machines based on traces analysis
- at our disposal
  - functioning traces [*partially observable* system]
  - formal model of a system : actions, laws, etc. [*partially descriptible* system]
  - theories to reason about the system model
- questions that can be addressed:
  - extract "novel" patterns: i.e. patterns that can not be explained by the formal model
  - extract "surprizing" patterns: i.e. patterns in traces that are conflicting the model

Epistemic Measure of Interestingness: an example II

PoC to finish

- implement of Balduccini action logic in ASP
- use of a very simple dynamical model (circuit + actions)
  - the cloud illustrates an unobservable part of the system



- objectif to show the potential of the approach

Other kind of reasoning

- Ontological reasoning is main target (but not easy to handle in clingo-ASP ... there are solutions in DLV)