Event Data Quality

Shaoxu Song

Tsinghua University
sxsong@tsinghua.edu.cn

Typical Event Data

Recording steps of achieving a certain goal

- Traditional enterprise office automation systems or scientific workflows
- Logs in Web services or online transactions

Often with very simple schema

- Sequences, of task names (event types)
- Simple graphs, with task names as labels

Event	Task Name	Timestamp	Prerequisite	...
t_1	order	2010.03.24.09.23.53	–	...
t_2	pay by c	2010.03.24.09.34.24	t_1	...
t_3	check	2010.03.25.07.32.25	t_1	...
t_4	re-order	2010.03.25.07.22.56	t_2, t_3	...
t_5	delivery	2010.03.26.09.24.42	t_4	...

Event Data Quality Issues

- Missing events
  E.g., delivered without any payment step
- Erroneous events
  E.g., pay by c, where c denotes cash or credit card?
- Duplicate events
  E.g., Email customer = Send notification

Inaccurate or duplicate events occur for various reasons

- Forgot to submit when manually recording event logs
- Suffered from system failures
- Mess after collecting events from heterogeneous execution environment
- ...

Improving Event Data Quality is Urgent

Without addressing inaccurate/duplicate events, applications and mining over event data are not reliable.

Provenance: steps used to produce some data

- It can be thought of as a graph which captures the prerequisite dependencies between entities involved in processes
- Queries of provenance as calculating transitive closures of prerequisite dependencies
  E.g., owing to missing events, no payment step is found before delivery

Mining: finding interesting patterns of event occurrence

- Inaccurate timestamps may distort the order of events
  E.g., is it possible to ship goods before payment
- Opaque event names prevent mining in heterogeneous sources
  E.g., can either Email Customer or Send Notification before delivery

Cleaning Event Data

- Recovering missing events
- Repairing dirty events
- Matching heterogeneous events

To perform data cleaning, we need some knowledge describing the truth of data

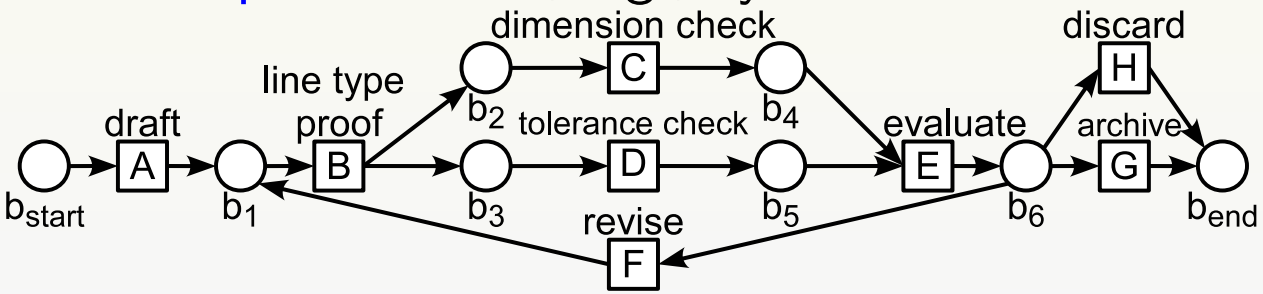
- Metadata, e.g., data types
  Helpless in identifying duplicate events
- Integrity constraints, e.g., functional dependencies
  Not powerful enough to express the semantics among events

Simple schema of event data does not contribute much in improving event data quality

Constraints on Event Data

Events do not occur randomly

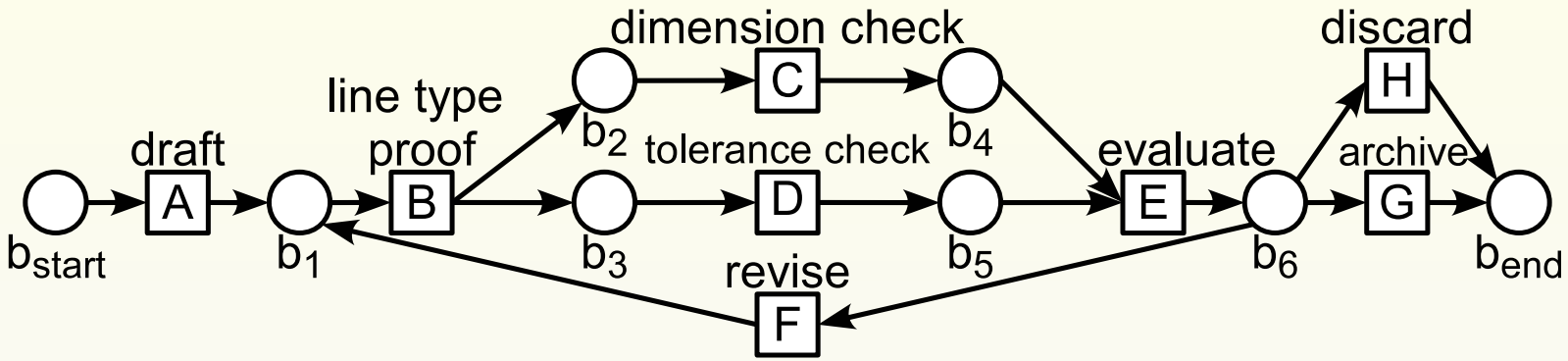
- Should obey certain discipline
- Process specifications, e.g., by Petri Net



- Event pattern, e.g., in Complex Event Processing (CEP)
  `SEQ(B, AND(C, D), E)`

Outline

Specification as Constraints



Each **square** (namely *transition*) denotes a task in the process specification, e.g., transition `A` represents a task of drafting

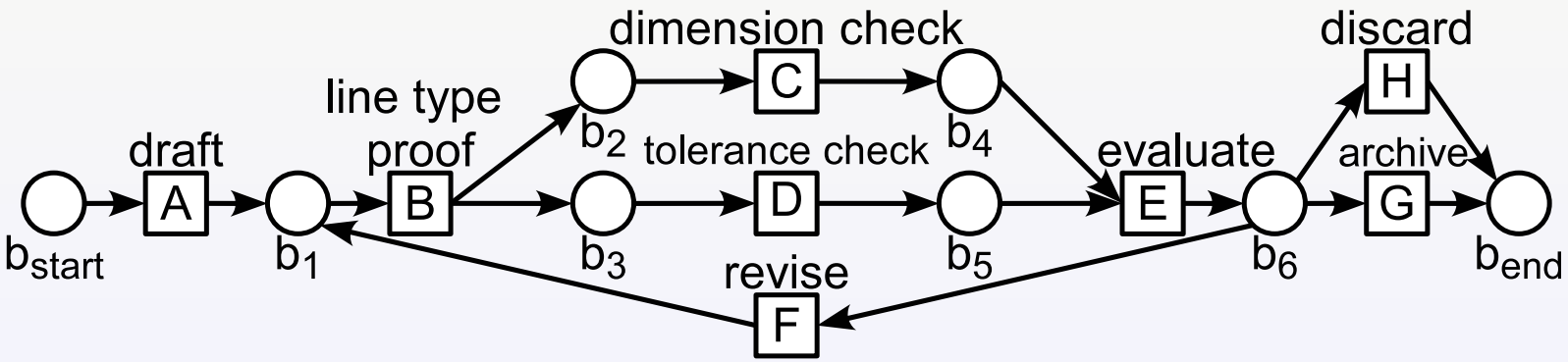
- All the arrows attached to a transition denotes the corresponding flows should be executed in parallel
- E.g., both the dimension checking (task `C`) and the tolerance checking (task `D`) should be conducted after line type proofing (task `B`) in the drawing.

Moreover, the process can carry on evaluating the drawing (task `E`) only if both `C` and `D` are accomplished.

Specification as Constraints

Circles are choice nodes, called *places*, which always appear between transitions

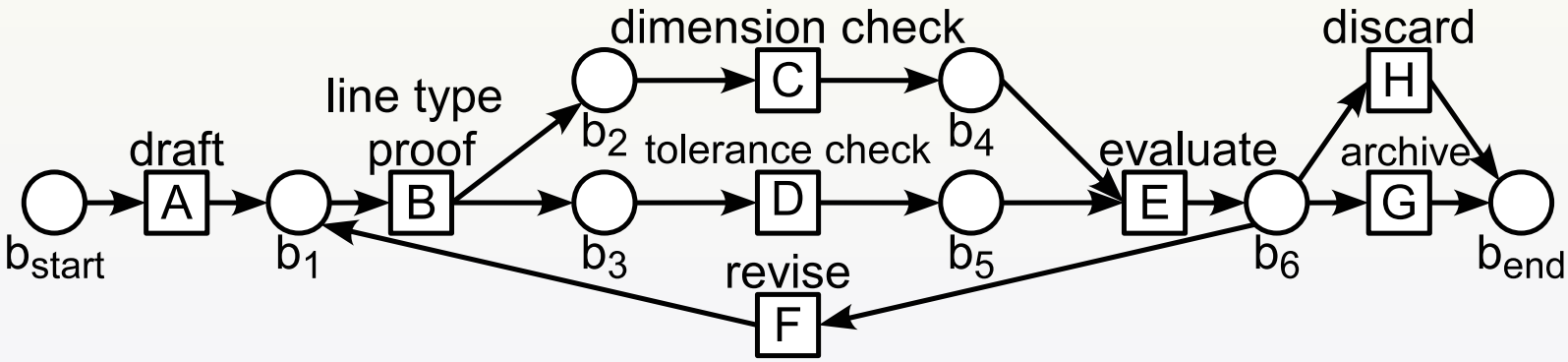
- It indicates that only one of the flows going out a place can be executed (XOR semantics)
- E.g., place b_6 leads to either revising the drawing (task `F`), archiving it (task `G`) or discarding it (task `H`) after evaluation (`E`)



Conformance

An execution of the process generates a sequence of events, where each event corresponds to a task in the process specification

- We say that a sequence **conforms** to the specification if it successfully executes from the source place b_{start} to the sink place b_{end}
  - Exactly following the flow constraints of parallel and choice in the specification

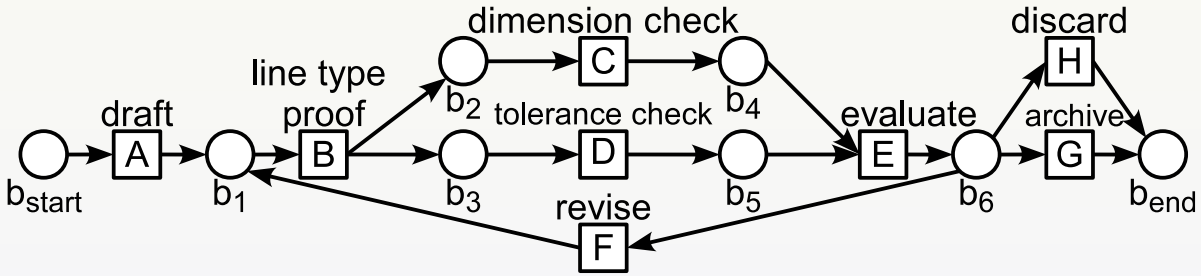


- <ABCDEG> denotes a complete execution of engineering drawing including steps `drafting`, `line type proofing`, `dimension checking`, `tolerance checking`, `evaluating`, `archiving` from b_{start} to b_{end}.

Missing Events

Owing to various data quality issues, event logs are often incomplete

- <ABCEG> has an event D missed during the collection of event logs from the database for dimension checking
- Will return erroneous provenance steps
- If missing events occur frequently, affect mining results



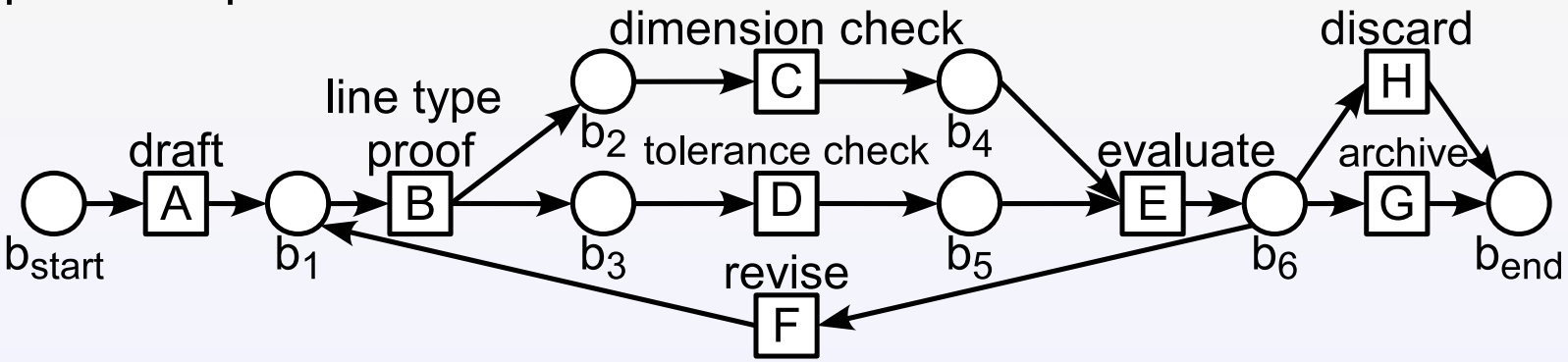
ID	Sequence of events
1	<A, B, C, D, E, G>
2	<A, B, C, E, G>
3	<A, B, C, D, G>

Possible Recoveries

Multiple recoveries exist for an incomplete sequence

- To recover the sequence
  <ABCDG>
- The results could be
  <ABCDEG>,
  <ABCDEFBCDEG>,
  <ABCDEFBCDEFBCDEG>, ...

Infinite sequences of events could be generated when loops exist in process specifications



Minimum Recovery

Following the **minimum change** discipline in improving data quality

- Also identify the optimal recovery of missing events that minimally differs from the original sequence.
- It is a rational assumption in improving data quality that people try to make the minimum mistakes, which is also applicable to missing events.

The minimum recovery guarantees to conclude the minimum number of events that are missing

- E.g., at least one event must be missing in the third sequence <ABCDG>

Hardness Analysis

Owing to choices and parallelization of flows, there may have vast alternatives to enumerate in the recovery

Generating the minimum recovery of missing events is indeed NP-hard

Theorem

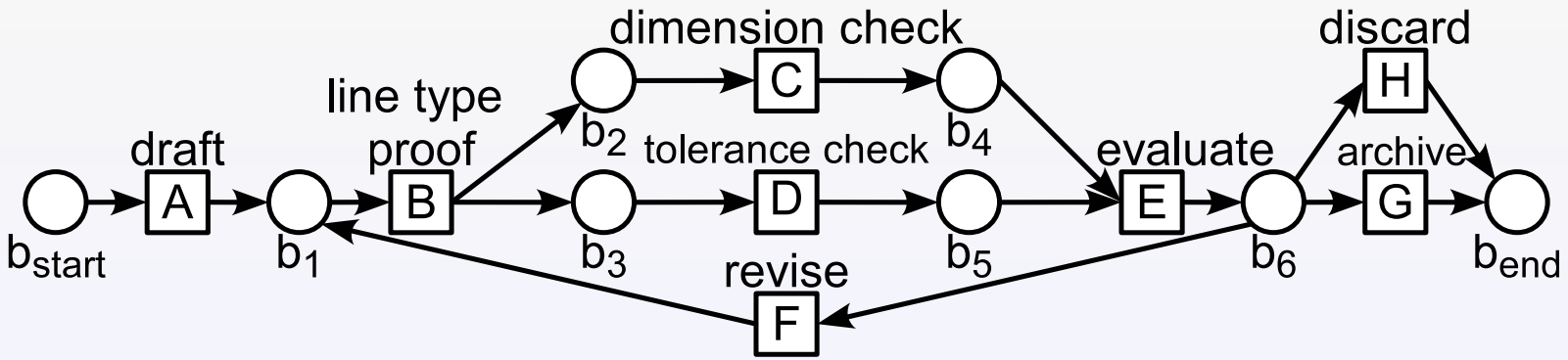
Given a sequence σ over a process specification $\mathcal{N}_s$ and a constant k, the problem is NP-complete to determine whether there exist a recovery σ' of σ such that $\sigma' \vDash \mathcal{N}_s$ and $\Delta(\sigma', \sigma) \leq k$.

- $\sigma' \vDash \mathcal{N}_s$ denotes that sequence σ' conforms to specification $\mathcal{N}_s$
- $\Delta(\sigma', \sigma)$ is the number of events recovered

Alignment Approach

To find the minimum recovery, the existing alignment approach [1]

- Studied in the business process management by enumerating all the valid sequences of events
- It falls short of efficiency owing to the redundancy in all possible event sequences
- E.g., to recover the sequence <ABCEG>, the results <ABCDEG> and <ABDCEG> have no difference w.r.t. the process specification, as C and D are executed in parallel after B and before E

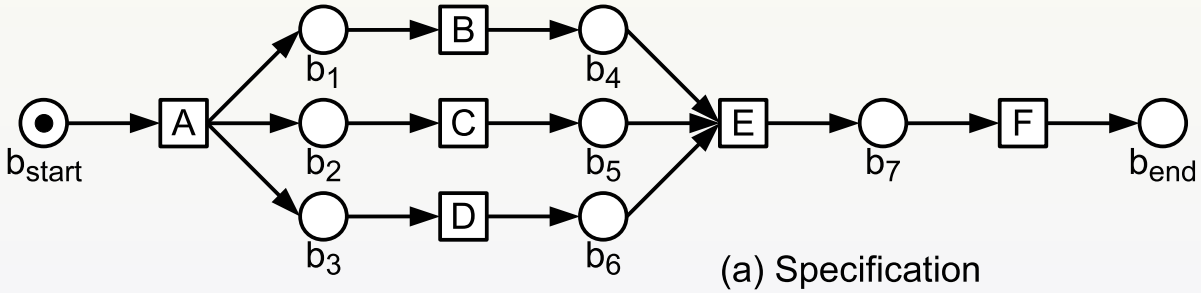


[1] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst. Aligning event logs and declarative process models for conformance checking. Business Process Management. 2012. 82-97.

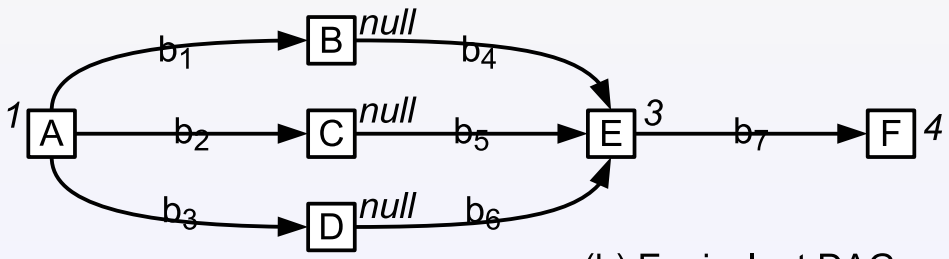
Branching Approach

Avoid unnecessary enumeration of parallel executions [2]

- For a parallel structure without any choice
- All topological sorts of the structure are valid executions, with the same number of events
- Any topological sort containing incomplete σ as a subsequence is a minimum recovery



(a) Specification



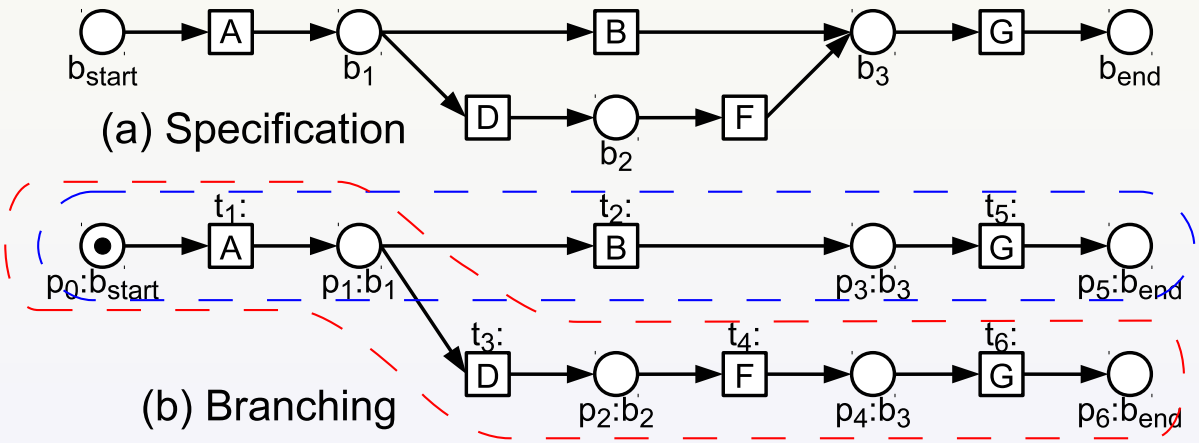
(b) Equivalent DAG

[2] J Wang, S Song, X Zhu, X Lin. Efficient Recovery of Missing Events. Proceedings of the VLDB Endowment 6 (10), 2013.

Branching Approach

Enumerate all possible parallel executions in a **branching net**

- Each branch corresponds to a parallel execution
- Branches may share similar contents
- Find the minimum recovery among all branches

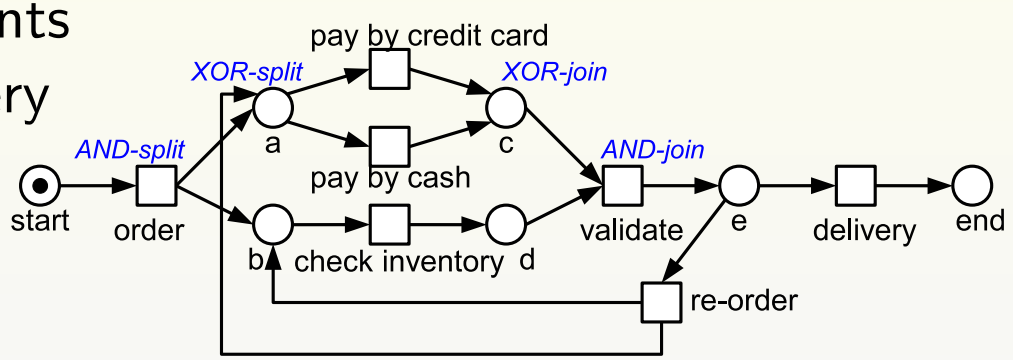


Outline

Dirty Data on Task Names

Syntactic errors: not in, but similar to values in the event domain
E.g., pay by c

Semantic errors: belongs to the event domain, but not denoting the true occurrence of events
E.g., re-order before delivery

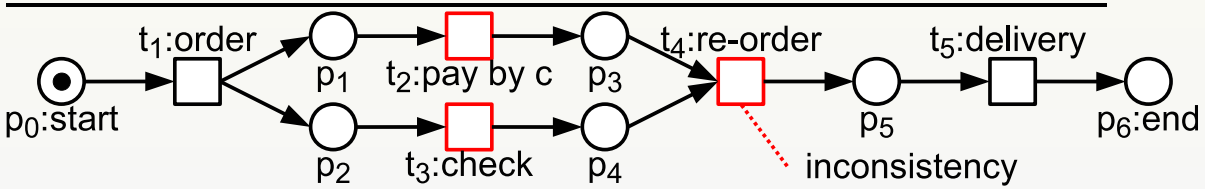


Event	Task Name	Prerequisite	Timestamp	...
t_1	order	–	2010.03.24.09.23.53	...
t_2	pay by c	t_1	2010.03.24.09.34.24	...
t_3	check	t_1	2010.03.25.07.32.25	...
t_4	re-order	t_2, t_3	2010.03.25.07.22.56	...
t_5	delivery	t_4	2010.03.26.09.24.42	...

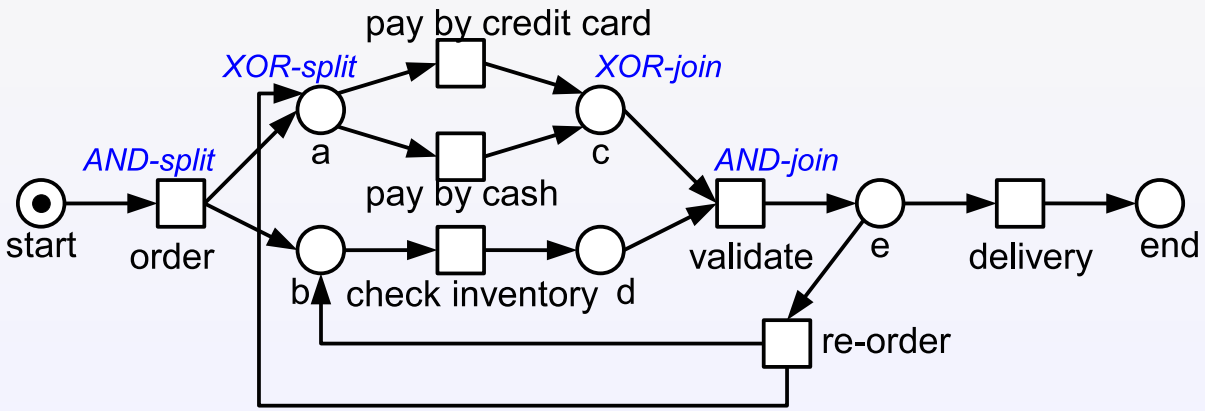
Repair of Task Names

- Consider event data as simple graphs rather than sequences, with prerequisite information

Event	Task Name	Prerequisite	Timestamp	...
t_1	order	–	2010.03.24.09.23.53	...
t_2	pay by c	t_1	2010.03.24.09.34.24	...
t_3	check	t_1	2010.03.25.07.32.25	...
t_4	re-order	t_2, t_3	2010.03.25.07.22.56	...
t_5	delivery	t_4	2010.03.26.09.24.42	...



- Modify task names w.r.t. conformance



Repairing Cost

$$\Delta(\pi, \pi') = \sum_{t \in T_\sigma} \delta(\pi(t), \pi'(t))$$

- $\pi'(t)$ is the new task name of event t in the repaired (N_σ, π')
- $\delta(\pi(t), \pi'(t))$ denotes the cost of modifying event t by $\pi'(t)$
  $\delta(\pi(t), \pi'(t)) = conf(t) \cdot dis(\pi(t), \pi'(t)) \cdot \frac{freq(\pi(t))}{freq(\pi'(t))}$
  - $conf(t)$ is the confidence associated to event t
    Higher cost to modify a high confidence event
  - $dis(\pi(t), \pi'(t))$ denotes the metric distance between two tasks $\pi(t)$ and $\pi'(t)$, e.g., edit distance on task names
  - $freq(\pi(t))$ and $freq(\pi'(t))$ are the frequencies of $\pi(t)$ and $\pi'(t)$, respectively, appearing in different execution traces in the log database
    E.g., suppose that "pay by credit card" is more frequent than "pay by cash"

Repairing Problem

To repair task names w.r.t. conformance by paying the minimum repairing cost

Theorem

The task name repairing problem is NP-hard.

Branch and Bound

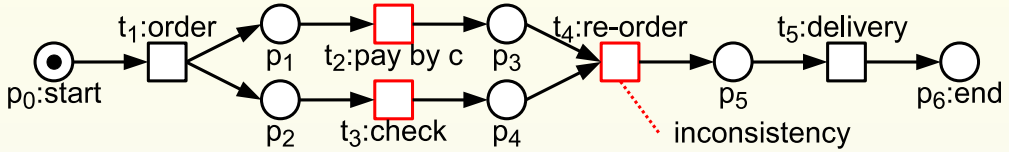
- partitioning possible repairs that can be generated
- pruning among partitions of possible repairs

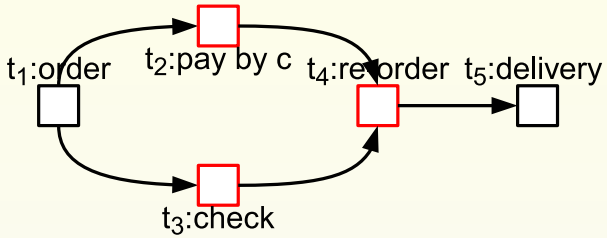
Key point: how to develop bounds of costs of possible repairs

Construct a conflict graph

- Vertices are events with a weight associated
  $w(t_i) = \min_{x \in T_s} \delta(\pi(t_i), x)$
  the minimum cost on all possible repairs of t_i
- Edges denote conflicts between events

Computing a Lower Bound





A lower bound of least cost

- The minimum weighted vertex cover of G with total weight $VC^*(G)$
- $VC^*(G) \leq \Delta(\pi, \pi')$ for any repair π'

However, computing the exact minimum vertex cover is still hard

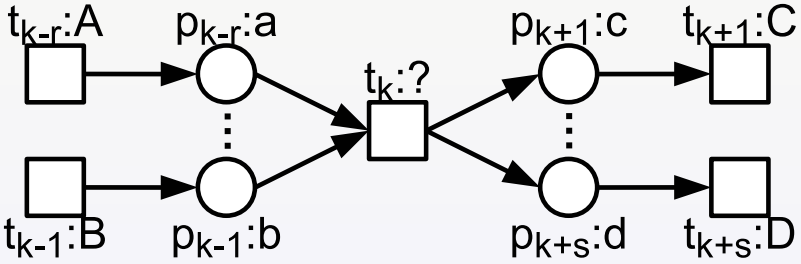
A loose but simple lower bound:

- Run approximation algorithm of VC by adding conflict edges into E until no conflicts left
- No edges in E share the same vertex
- As each edge should be covered by at least one vertex from the minimum vertex cover
- $\sum_{(t_i,t_j)\in E} \min\{w(t_i), w(t_j)\} \leq VC^*(G)$

One Pass Greedy Repair

To support fast approximate repairing by one pass through the events

- Scan events in the order of timestamps
- Check prerequisite conformance of each event
- If conflict is detected, greedily find a repair that
  - eliminate prerequisite conflicts of the event
  - introduce the least conflicts after the event



It is possible that no valid repair can be generated

Open Research Issues

Reasoning about time constraints

- **Consistency problem**: whether a given set of time constraints have conflicts
  Exist a nonempty instance of events that satisfies all the given time constraints
- **Implication problem**: whether a time constraint can be implied by others
  Remove redundant time constraints

Consistent Query Answering over Event Data

- E.g., provenance query
- Rather than answering over a single (minimum) repair/recovery
- Return those answers appearing in all possible (minimum) recoveries

Outline

Event Matching

Duplicate events, describing the same real world business activities

- Often exist in different divisions of a corporation
- In alike business processes among different companies

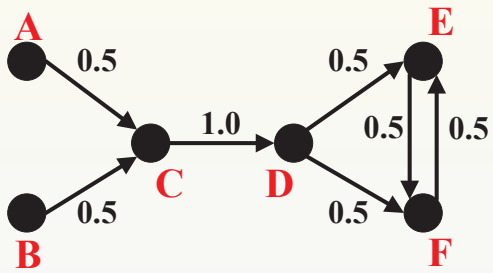
To find mapping between heterogeneous events

ID	Trace
1	Paid by Cash (A) → Check Inventory (C) → Validate (D) → Ship Goods (E) → Email Customer (F)
2	Paid by Credit Card (B) → Check Inventory (C) → Validate (D) → Email Customer (F) → Ship Goods (E)
...	.....

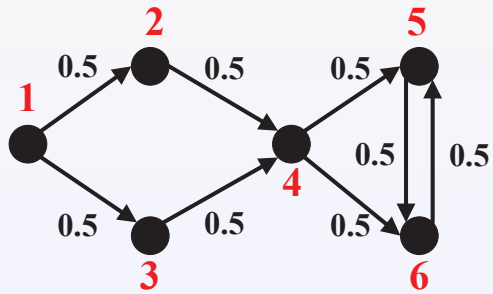
(a) Event log $\mathcal{L}_1$

ID	Trace
1	Order Accepted (1) → Paid by Cash (2) → Inventory Checking & Validation (4) → ????????? (5) → Send Notification (6)
2	Order Accepted (1) → Paid by Credit Card (3) → Inventory Checking & Validation (4) → Send Notification (6) → ????????? (5)
...	.....

(b) Event log $\mathcal{L}_2$



(c) Dependency Graph G_1 of $\mathcal{L}_1$



(d) Dependency Graph G_2 of $\mathcal{L}_1$

Matching Techniques

Syntactic: based on similar task names

- With the same name of "Paid by Cash"
- Opaque name problem
  - E.g., Email Customer vs. Send Notification
  - ??????? with encoding problem

Structural: based on similar neighbors (prerequisites and successors)

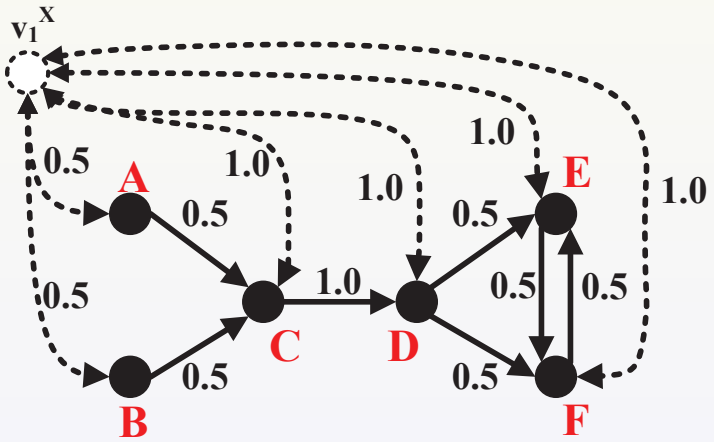
- Construct a graph to denote relationships between events, e.g., consecutive occurrence
- Employ graph matching techniques, such as Graph Edit Distance [3]
- Partial mapping problem
  - There is an extra "Order Accepted" step in $\mathcal{L}_2$ which is not considered in $\mathcal{L}_1$

[3] R. Dijkman, M. Dumas, L. García-Bañuelos: Graph Matching Algorithms for Business Process Model Similarity Search. BPM 2009: 48-63

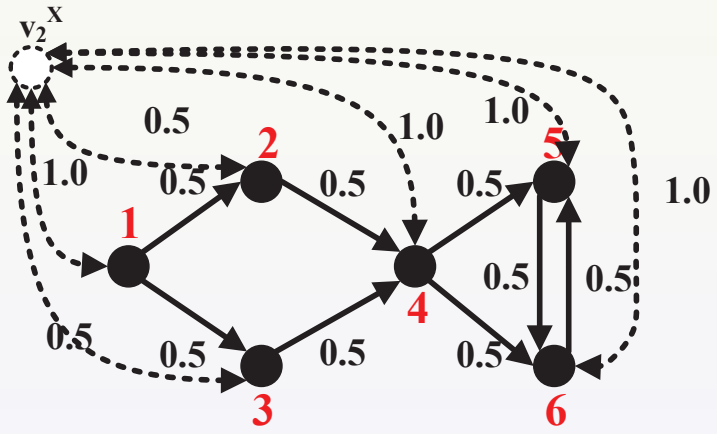
Capture Partial Matching

Intuition: Any event v could be the start/end event in the other side of matching

- add a virtual event v^X
- directly connecting to all the events as both prerequisites and successors



(a) Dependency Graph G_1 of $\mathcal{L}_1$



(b) Dependency Graph G_2 of $\mathcal{L}_1$

Now, each event, as possible start/end events, share at least some similarity (a common virtual event)

Event Similarity Function

SimRank like iteration similarity evaluation for event data

Initialization

- Rather than arbitrary assignment [4] , without consideration of partial mapping
- The similarity between two virtual events are initialized and fixed to 1 in iteration
- The similarity between other events are initialized to 0

Iteration: iteratively updates pair-wise similarity

$\mathcal{S}^n(v_1, v_2) = \frac{1}{2}(s^n(v_1, v_2) + s^n(v_2, v_1))$

$$s^n(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2') \mathcal{S}^{n-1}(v_1', v_2') \quad (1)$$

$$s^n(v_2, v_1) = \frac{1}{|\bullet v_2|} \sum_{v_2' \in \bullet v_2} \max_{v_1' \in \bullet v_1} C(v_1, v_1', v_2, v_2') \mathcal{S}^{n-1}(v_1', v_2') \quad (2)$$

[4] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, P. Zave: Matching and Merging of Statecharts Specifications. ICSE 2007: 54-64

Properties and Concerns

Convergence

Theorem

For all $v_1 \in V_1, v_2 \in V_2, \lim_{n\to\infty} \mathcal{S}^n(v_1, v_2) = \mathcal{S}(v_1, v_2)$.

Special events with early convergence

Let $l(v)$ denotes the longest distance from v^X to v (could be ∞ if loops exist from v^X to v).

Proposition

For any two events $v_1 \in V_1$ *and* $v_2 \in V_2$,
$\mathcal{S}^{h+1}(v_1, v_2) - \mathcal{S}^h(v_1, v_2) = 0$, *for all* $h \geq \min(l(v_1), l(v_2))$.

- Similarities of some node pairs are guaranteed to converge in a certain number (say h) of iterations
- Safely prune the updates of those similarity pairs with early convergence after h iterations

Integrating with Other Similarities

Forward and **backward** similarity

- Have given formulas of prerequisite (in-neighbors) similarities
- Symmetrically consider backward similarities on successors (out-neighbors)

Consider other similarity as prior knowledge, in initial assignment

- E.g., syntactic similarity on task names
- Update similarity only when having further evidence, i.e., iteration similarity is greater than initialization
- The conclusion of convergence still holds

Outline

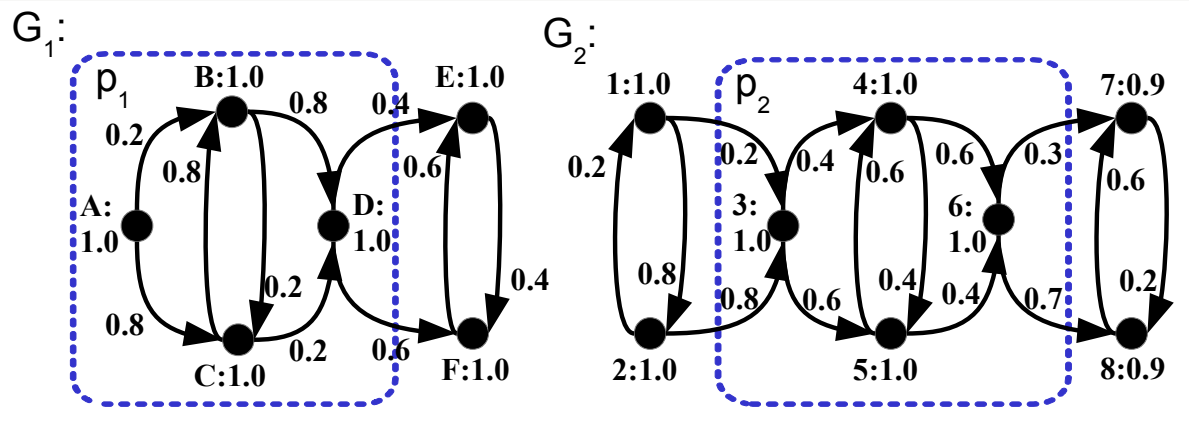
Matching with Patterns

Patterns: things often occur like this

- e.g., in Complex Event Processing (CEP)
- `SEQ(Received, AND(Payment, Inventory), Shipped)`

For any mapping

- given a pattern in one side
- to investigate whether it happens similarly in the other side
- if sharing similar pattern frequency, the mapping is probably reliable [5]



[5] X. Zhu, S. Song, J. Wang, P. S. Yu, J. Sun: Matching Heterogeneous Events with Patterns. ICDE 2014

Matching with Patterns

More discriminative

- Each node is a singleton pattern
  But D shares the same frequency 1.0 with many other events
- Edges are special patterns with two events, in a simple graph
  Does D=3? as they have similar in/out edges
- General patterns, as hyper edge in a hyper graph
  D=6!

Consider all possible mappings

- compute pattern based similarity for each mapping
- return mappings with higher similarities

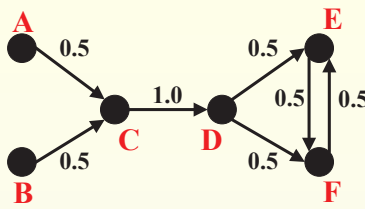
To enable pruning among possible mappings

- we develop bounds of similarity scores with patterns

Open Research Issues

ID	Trace
1	Paid by Cash (A) → Check Inventory (C) → Validate (D) → Ship Goods (E) → Email Customer (F)
2	Paid by Credit Card (B) → Check Inventory (C) → Validate (D) → Email Customer (F) → Ship Goods (E)
...	.....

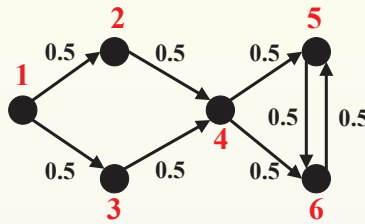
(a) Event log $\mathcal{L}_1$



(c) Dependency Graph G_1 of $\mathcal{L}_1$

ID	Trace
1	Order Accepted (1) → Paid by Cash (2) → Inventory Checking & Validation (4) → ????????? (5) → Send Notification (6)
2	Order Accepted (1) → Paid by Credit Card (3) → Inventory Checking & Validation (4) → Send Notification (6) → ???????? (5)
...	.....

(b) Event log $\mathcal{L}_2$



(d) Dependency Graph G_2 of $\mathcal{L}_1$

Matching composite events

- C+D corresponds to 4
- Rather than predefined patterns
- Composite events need to be determined w.r.t. better matching

- **Evaluation**: how to define better matching among composite events
- **Complexity**: various combinations of composite events and matching

Summary

Addressing missing/inaccurate events

- Following the minimum change principle in data quality
- With process specification as constraints
- Time constraints

Identifying duplicate events

- By graph matching techniques
- With the consideration of partial mapping
- Using patterns as features

Interesting research directions

- Reasoning about constraints over event data
- Consistent query answering over inconsistent event data
- Matching composite events