BPM 2005 - Third International Conference on Business Process Management

Presentation by

Domenico Saccà

ICAR-CNR &University of Calabria

Mining Hierarchies of Models: From Abstract Views to Concrete Specifications

Gianluigi Greco[1], Antonella Guzzo[2], Luigi Pontieri[2]

([1]) Dept. of Mathematics, University of Calabria, Italy

([2]) ICAR-CNR, National Research Council, Italy

Outline

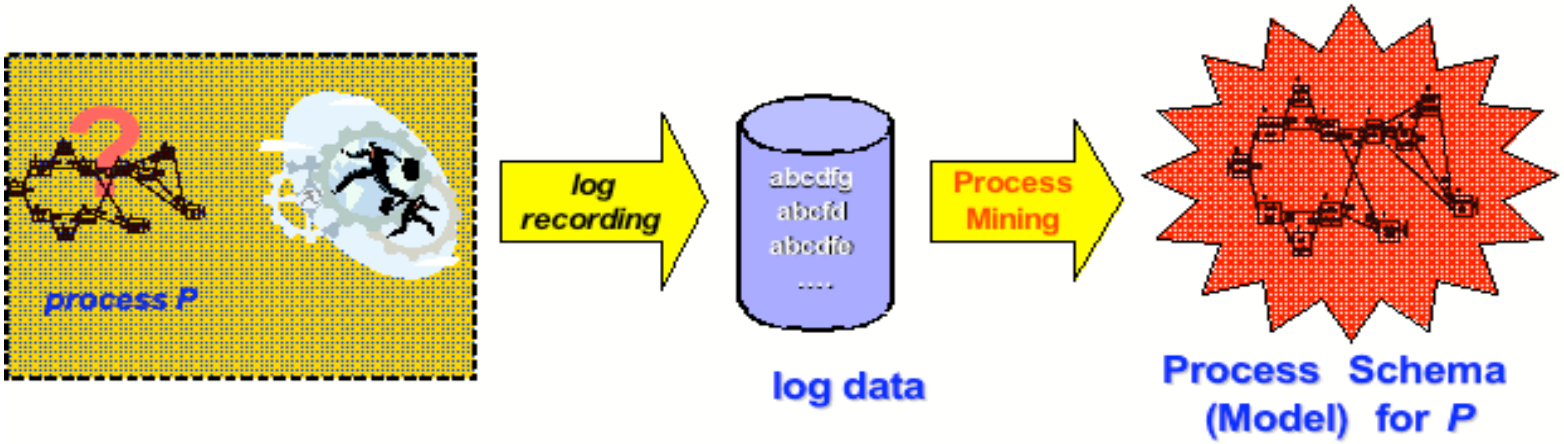
- **Prologue**
- ***Phase 1***: Mining a hierarchy of workflow schemas

  (based on [*Greco, Guzzo, Pontieri & Saccà,* 04])
- ***Phase 2***: Restructuring a schema hierarchy via abstraction
  - A basic framework for workflow abstraction
  - The restructuring algorithm
- **Concluding remarks**

Prologue (1/3):

Process Mining

- Process Mining aims to automatically discover a model for a process, based on data gathered during its past enactments



- Why process mining?
  - Mined models helps to better comprehend the process behavior, and to (re-)design/optimize concrete workflow models
  - Modeling complex processes is a difficult and expensive task

Prologue (2/3):

Motivation: Mining processes with complex behavior

- *Problem*: complex processes may involve lots of activities, and complex behavioral rules for combining them
  - the discovered model may fail in representing the process with enough accuracy
  - … and may be too complex for business users who want to monitor and analyze process executions at an appropriate abstraction level

Desiderata

Execution Classification

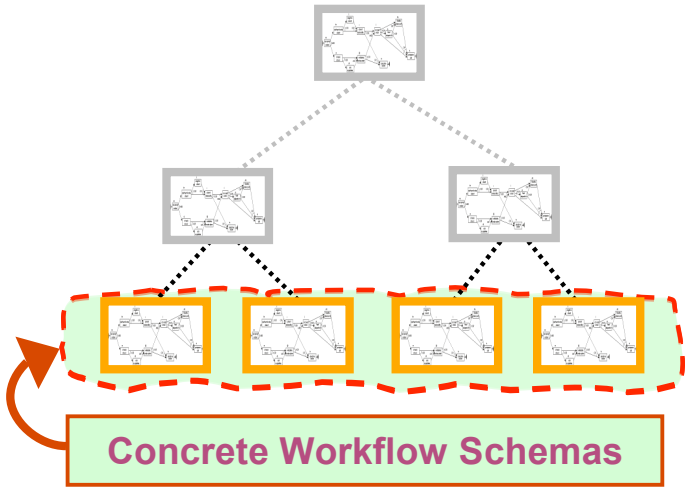
This allows to gain in accuracy,modularity and understandability, w.r.t. a single workflow schema mixing all executions

Abstraction

BPA platforms (e.g, iBOM by HP) allow to manually define abstract views over a workflow, by mainly aggregating groups of activities

The proposed approach in a nutshell

- Discover an expressive and easy to understand process model, consisting of a tree of workflow schemas



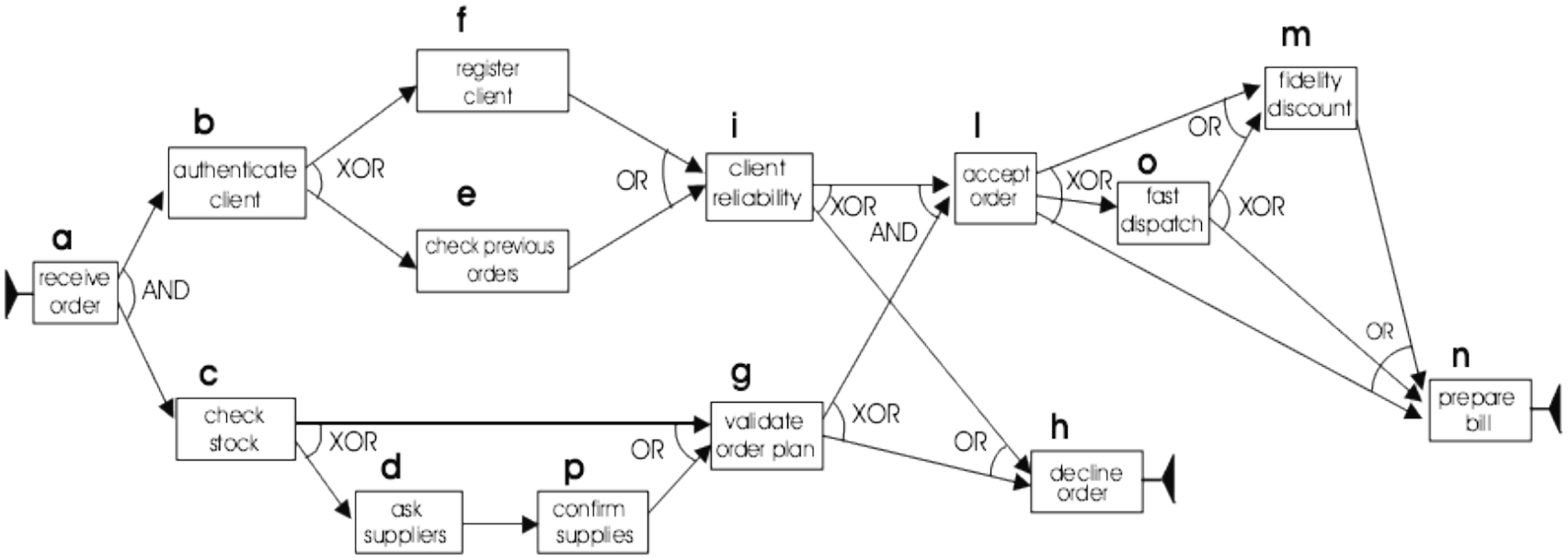
- The tree describes the process behavior at different level of details
- At the highest level of detail (leaves of the tree), the schemas could be used to support the design of concrete workflow models
- At lower levels, the schemas are abstract views over heterogeneous behaviors, which could support analysis and monitoring tasks

- The technique: A two-phase approach, combining mining strategies and abstraction methods
  - First, we mine a tree of workflow schemas, by using a hierarchical, top-down, clustering algorithm
  - Then, the mined model is restructured at several levels of abstraction, in a bottom-up way (i.e., from the leaves to the root)

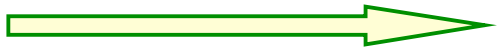
Outline

- Prologue
- ***Phase 1***: Mining a hierarchy of workflow schemas
- ***Phase 2***: Restructuring a schema hierarchy via abstraction
  - A basic framework for workflow abstraction
  - The restructuring algorithm
- Concluding remarks

Workflow schemas and logs (by example)

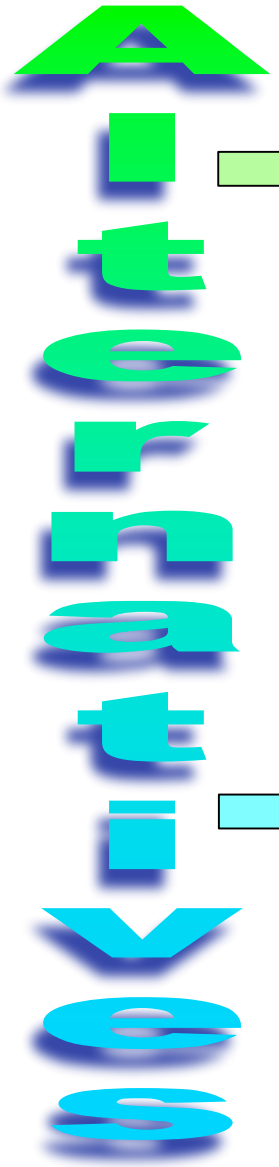


- Instance of process *P*:
  - A connected sub-graph of the control flow graph of P, containing at least the starting activity and one final activity, and compliant with all constraints
- Trace of *P*:
  - A sequence of task IDs corresponding a topological ordering of an instance of *P*
- Log of *P*:
  - A set of traces of *P*



```
abficgln,
acbidpegln
abficdgh
```

How to mine accurate models?



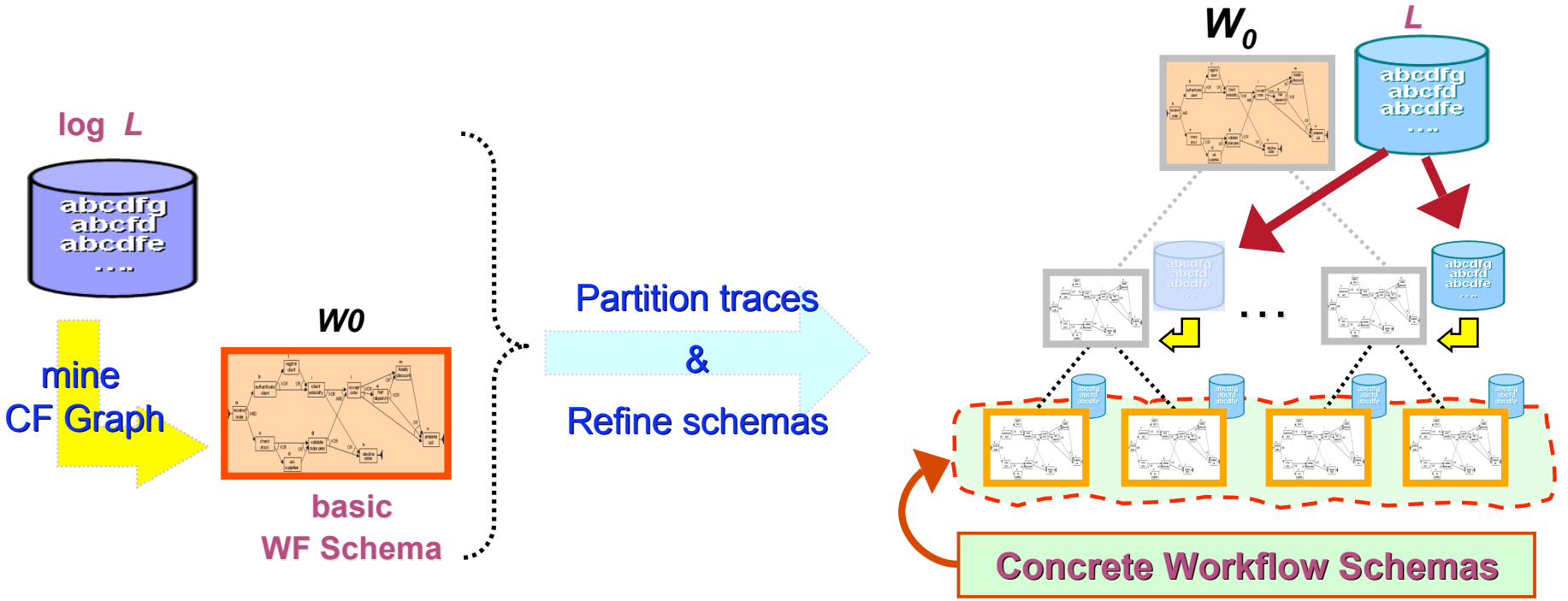
Use more expressive languages / meta-models

- e.g., control flow graphs could be enriched with additional "global" constraints, relating nodes that are not adjacent to each other
- but, explicitly handling such constraints may lead to knotty models and makes harder the process mining task

Mine different schemas (usage scenarios)

- Complex behavioral rules can be caught indirectly, by recognizing different unexpected and frequent behavioral patterns
  - unexpected w.r.t. a given control flow graph, but frequent in the log
  - such patterns evidence the existence of constraints (or usage patterns) that are not properly modeled by the graph
- Use a set of workflow schemas
  - more expressive, and accurate, than a single schema
  - but still intuitive and easy to mine

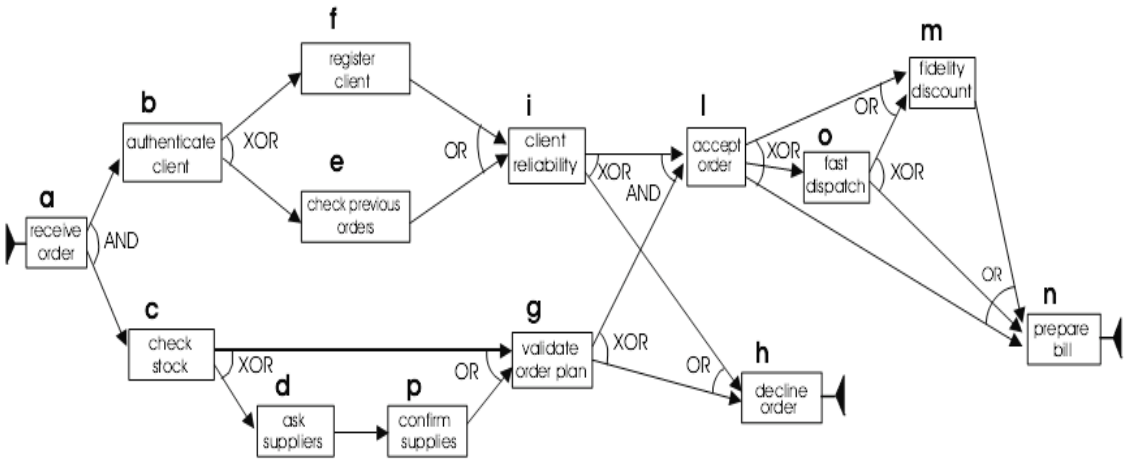
A clustering-based approach to Process Mining



- The basic schema W_0 is a first attempt to model all the log traces
- Iteratively, a leaf schema is refined to get higher **soundness**
  - ***Soundness*** = % of traces of the leaf schemas that occur in the log
  - the associated traces are split into (more homogenous) clusters
  - a new schema is derived for each cluster
- The schemas in the tree (specifically its leaves) represent a model sounder than W_0

The first schema induced

- Preliminary schema induced: W_0



- W_0 coincides with the original schema
  - it does not model the additional constraints
- W_0 hence admits "extraneous" traces
  - e.g., **acgbfilmn**

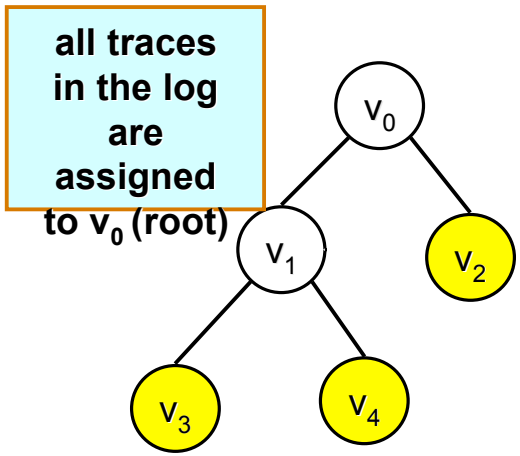
- In order to get higher soundness, W_0 we search for clusters of traces that correspond to different usage scenarios
- To this aim a set of discriminating features is extracted:
  - $\phi_1 : [\,f\ i\ l\,] \not\rightarrow m$

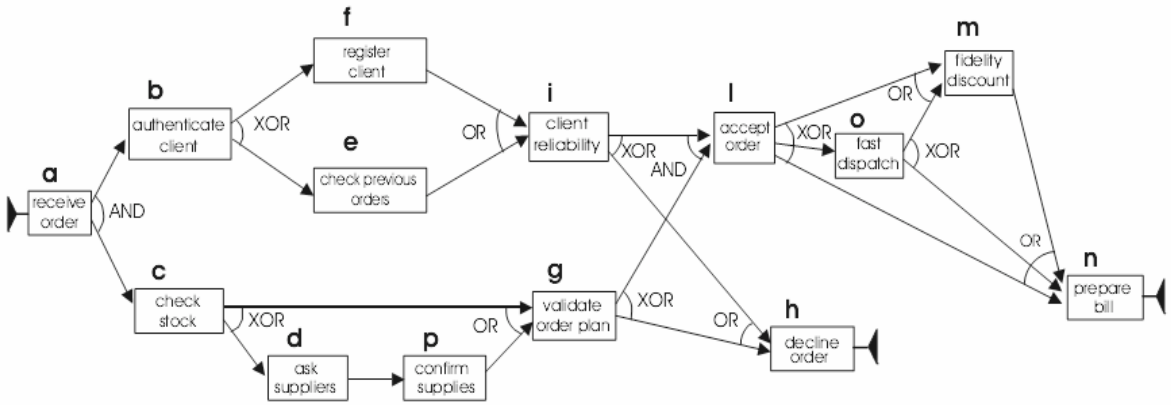
    Fidelity discounts are never applied on new (just registered) clients
  - $\phi_2 : [\,d\ g\ l\,] \not\rightarrow o$

    If external supplies have been checked, no fast dispatch occurs

The discovered hierarchy of schemas

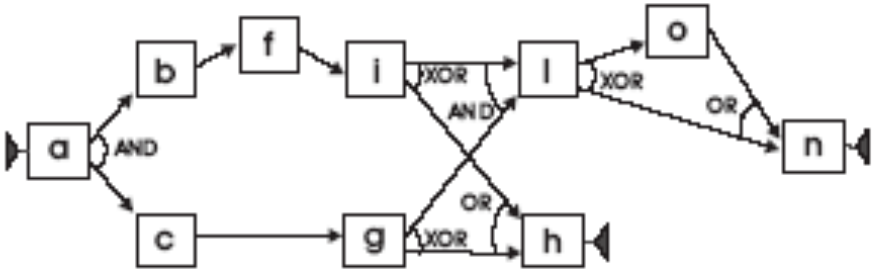
- Schema hierarchy obtained with *k*=2, *maxSize*=5, and γ=0.85



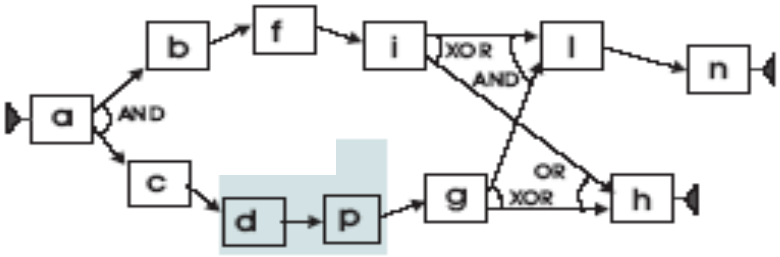


Workflow schema W_0 for node v_0

W_0 must be refined because its soundness is not high enough



Workflow schema W_3 for node v_3



Workflow schema W_4 for node v_4

- the leaf schemas (the only ones shown here) constitute, as a whole, a maximally sound and complete disjunctive scheme

Outline

- Prologue
- Phase 1: Mining a hierarchy of workflow schemas
- Phase 2: Restructuring a schema hierarchy via abstraction
  - A basic framework for workflow abstraction
  - The restructuring algorithm
- Concluding remarks

From a hierarchy to a taxonomy of schemas

- The restructuring phase is meant to produce a sort of *taxonomy* modeling all process variants discovered in the mining phase
  - The taxonomy provides for different abstraction levels
  - The taxonomy is more readable and usable than a flat model
- What a kind of generalization notion?
  - We adopt a very simple notion based on abstraction relationships between the involved activities, which is meant to support the derivation of abstract views over different workflows schemas
  - It is not a notion of dynamic inheritance, ensuring properties of behavioral consistency
  - …

Generalization of workflow schemas

- Given two workflow schemas *W* and *W'* (with activity set *A* and *A'*, resp.), we say that *W generalizes W'*, denoted by $W' \prec W$, if :
  1. for any activity *x* in *A* either *A'* contains *x* or there exists at least one activity *y* in *A'* such that *x* "*abstracts*" *y*, and
  2. there is no activity in *A'* that "*abstracts*" *x*
- According to this notion we define schema taxonomies

  A schema hierarchy *H* for *P* is a *schema taxonomy* if *Schema*(v) $\prec$ *Schema*(v') for any *v, v'* such that *v'* is a child of *v*

Abstraction relationships among activities

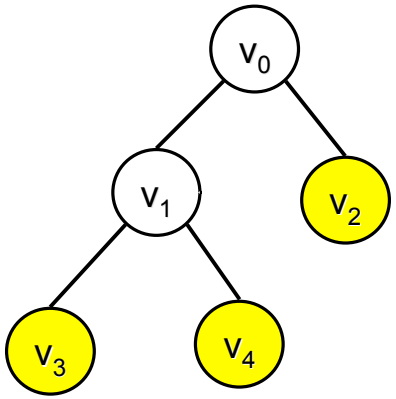
- Basic relationships: abstraction dictionary $D=<Isa, PartOf>$
  - $(b, a) \in Isa$ means that b is a refinement of a
  - $(b, a) \in PartOf$ means that b is a component of a
- Derived relationships
  - a ***implies*** a' w.r.t. D, denoted by $a \rightarrow^D a'$, if
    - $(a', a) \in D.Isa$, or
    - $(a', a) \in D.PartOf$, or
    - (recursively) there exists an activity x such that $a \rightarrow^D x$ and $x \rightarrow^D a'$
  - The set of activities implied by a w.r.t. D is referred to as $impl^D(a)$
- Complex activities
  - An activity a is *complex* if $impl^D(a)$ is not empty
  - It is a higher level concept defined over the (basic) activities that actually occur in the executions

Outline

- Prologue
- Phase 1: Mining a hierarchy of workflow schemas
  - Process Mining problem: formal framework
  - A clustering-based algorithm for Process Mining
- Phase 2: Restructuring a schema hierarchy via abstraction
  - A basic framework for workflow abstraction
  - The restructuring algorithm
- Concluding remarks

Restructuring a schema hierarchy

- Every non-leaf schema in the hierarchy is replaced with an abstract schema that generalizes those of its children
  - The process is applied in a bottom-up way, i.e., form the leaves to the root of the hierarchy

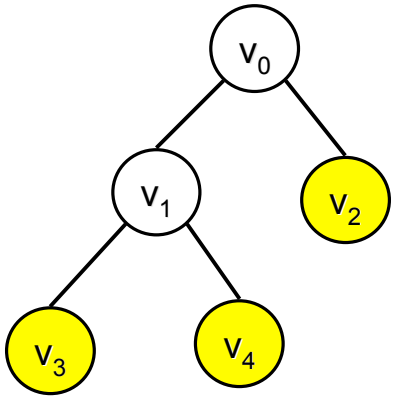


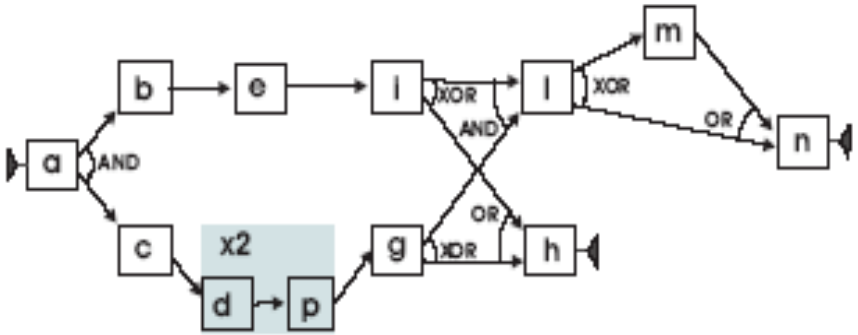
Restructuring a schema hierarchy

- Computation of the generalized schema for a non-leaf node
  - For each child schema *abstract* "specific" activities (activities that do not occurring in all children)
  - Merge all the children schemas into a single one
    - compute the union of the graphs, and adjust all constraints
  - *Abstract* "specific" activities appearing in the merged schema
- Abstracting "specific" activities
  - Only activities appearing in all children are surely kept in the generalized schema, while remaining ones, are abstracted
    - A group of "specific" activities is replaced with a complex activity that implies them all via IS-A or PART-OF relationships
  - We need a strategy to recognize groups of "specific" activities that can be abstracted by the same higher-level activity ….

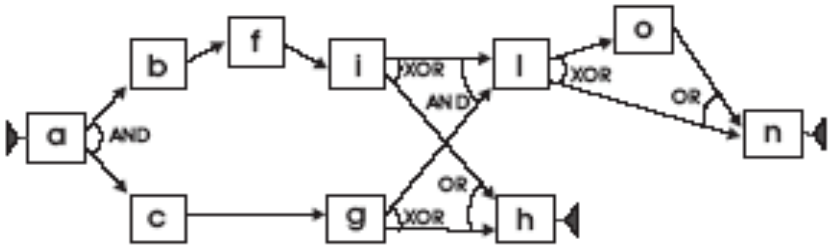
The mined schema hierarchy

- The hierarchy of workflow schemas extracted so far

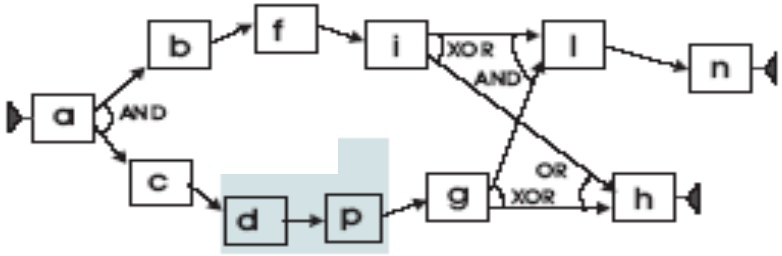




Workflow schema W_2 for node v_2



Workflow schema W_3 for node v_3

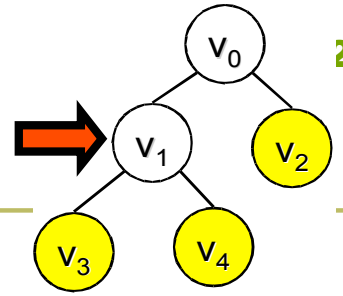


Workflow schema W_4 for node v_4

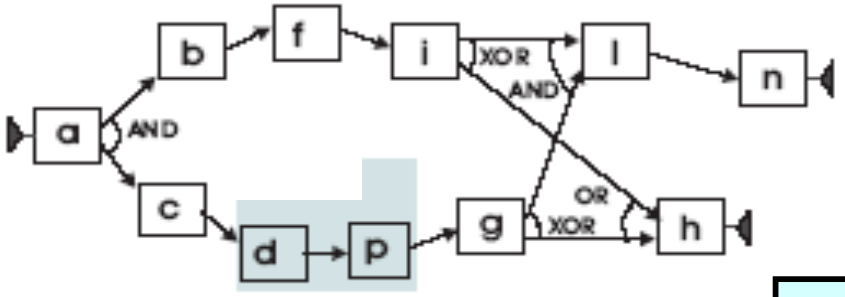
- … can be transformed into a taxonomy, by restructuring the schemas of all non-leaf nodes, $\mathbf{v_1}$ and $\mathbf{v_0}$, in a bottom-up fashion

The approach in action:

Restructuring a schema hierarchy

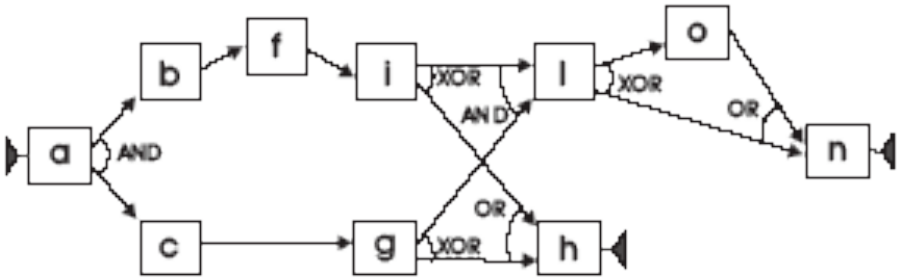


Schema of v_3

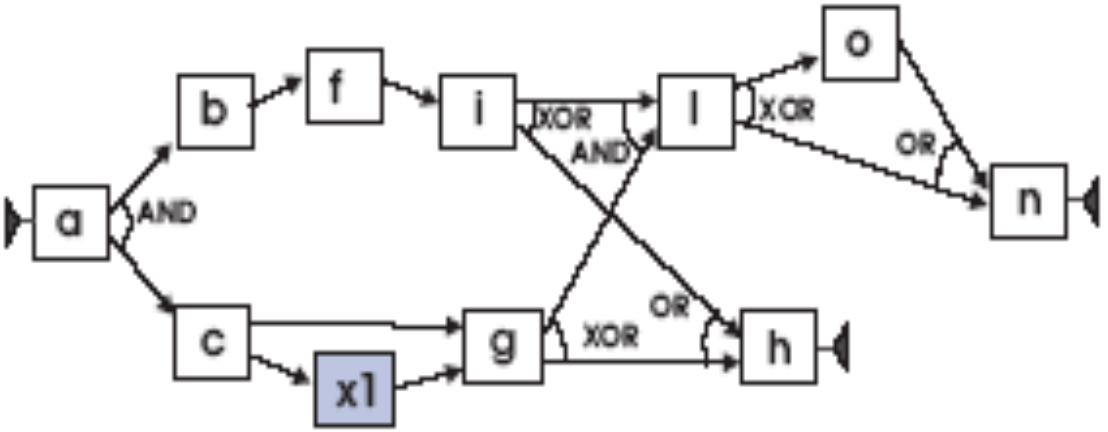


union

Schema of v_4



Generalized schema for v_1



Abstraction Disctionary

(assumed initially empty)

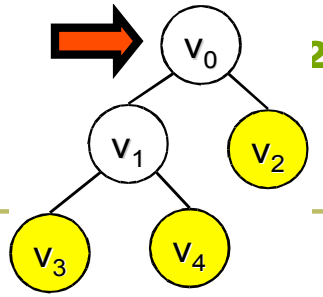
PART-OF **=**
{(d,x1), (p,x1) }

ISA **= { }**

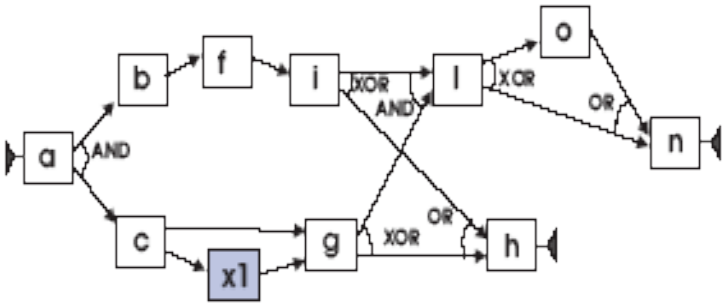
This is the only merge-s[illegible],
which are abstracted into activity **x1**, via PART-OF

The approach in action:

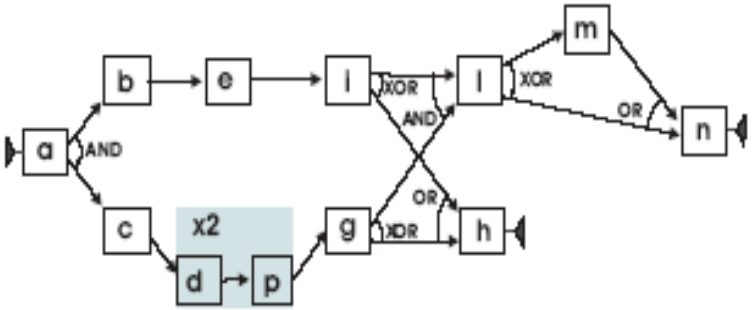
Restructuring a schema hierarchy



generalized schema of node v_1



schema of node v_2

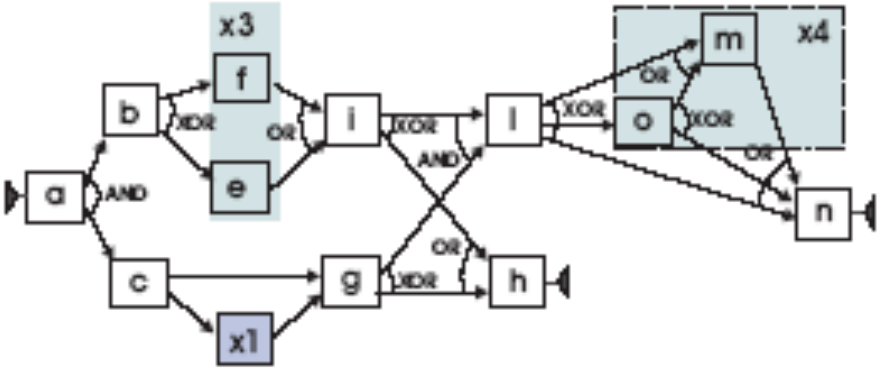


PART-OF = {(d,x1), (p,x1)}
ISA = { }

x2 contains the same basic activities as x1 (according to the dictionary)

therefore it is merged into x1 (no new activity is created)

generalized schema of root v_0



PART-OF =
{ (d,x1), (p,x1),
(f,x3), (e,x3),
(o,x4), (m,x4) }

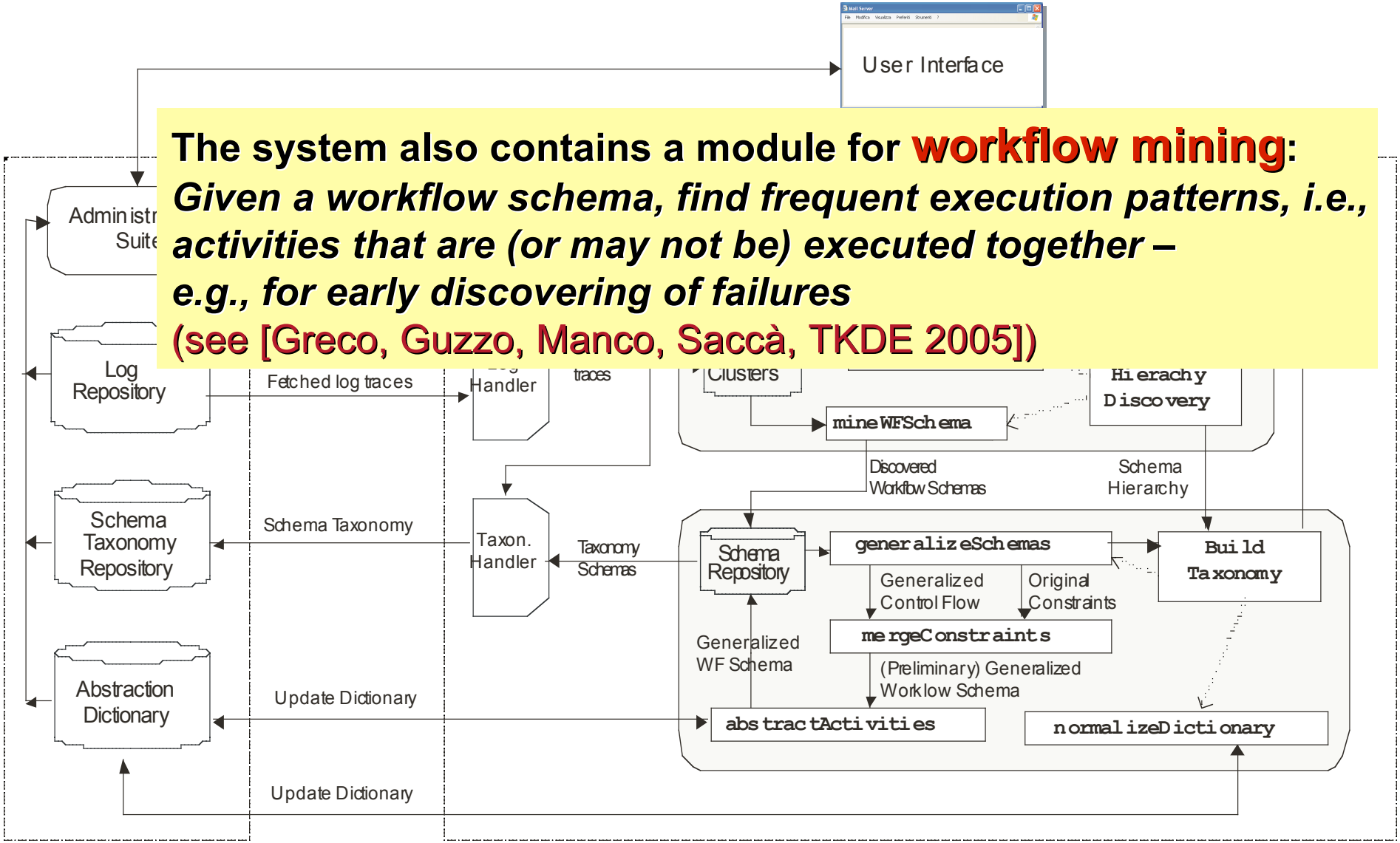
ISA = { }

Outline

- Prologue
- Phase 1: Mining a hierarchy of workflow schemas
- Phase 2: Restructuring a schema hierarchy by abstraction
  - A simple abstraction framework for activity
  - The generalization algorithm
  - Measures for selecting activities to merge
- Concluding remarks

A system for mining expressive process models

The system, developed in Java, integrates the algorithms illustrated previously



The system also contains a module for workflow mining:
Given a workflow schema, find frequent execution patterns, i.e., activities that are (or may not be) executed together – e.g., for early discovering of failures
(see [Greco, Guzzo, Manco, Saccà, TKDE 2005])

Conclusions

- A new kind of process modeling: hierarchy (taxonomy) of graph-based workflow schemas
  - can accurately model executions ruled by expressive specification models or by complex behavioral rules
  - The process is described modularly, at different level of details
- A (greedy) algorithm for mining a hierarchical model
  - The algorithm produces a tree of schemas, which is expanded until a maximal level of soundness is reached (under size limitations)
  - Experimental results on several synthetic datasets prove the effectiveness and scalability of the approach
- A technique for restructuring non-leaf schemas via abstraction
  - A greedy pair-wise approach
  - Adding semantics to intermediate nodes

Extensions (current and future work)

- Exploiting richer formats for log traces
  - Different kinds of events might be recorded for any task (start, end, termination, abort)
  - Information on the context of execution (executors/services, manipulated data values, …)
- Integrating the technique within a thorough analysis environment
  - Supporting the analysis and (re-)design (or customization) of processes, as well as their optimized enactment
  - Extending abstraction mechanisms
  - Including OLAP tools
- Opening towards Process Ontologies
  - PM for supporting the definition of new taxonomies
  - Ontologies as background knowledge guiding process mining and abstraction

Thank you

On behalf of the authors: Gianluigi, Antonella and Luigi