Business process learning for real time enterprises
Problem

Discover a process definition automatically from a “small” set of process traces and domain knowledge.

(e.g. a set is small if the set size is not sufficient for statistical machine learning).

Practical needs:

- In business problem domain – change a process by a deadline (real time learning); the learned process itself can be real time
- In software development domain – provide a process to ensure consistency of artifacts, integrate tools, orchestrate human activities, ensure repeatability
- In manufacturing – describe a method of producing a product on an assembly line that integrates software, hardware automation and work by humans
Process learning alternatives

Related work

- Discovery of a process from comprehensive event logs (van der Aalst, Alex Wolf, Jonathan Cook ..)
- Traces must provided full coverage according to some criterion (e.g. at least branches)
- Too many traces are needed ....
- Relying only on traces may overconstrain learned process definition

Our Approach

- Utilize various kinds of domain knowledge
- Utilize a small set of execution traces to verify learned processes
Definitions of key terms

- Process (definition) – a specification of a systematic way to provide a product or a service

- Process execution – instantiation of a process

- Resource – an entity used or produced by a process

- Artifact – a resource utilized by an activity (input) or produced by an activity (output product)
What defines a Process?

What must be described for a process?

1. Set of activities
2. Activity – interface, agent, relations to other activities, pre- and post-conditions, resources
3. Artifacts
4. Control flow (branching, iteration/recursion)
5. Data flow
6. Constraints (e.g. real time)

Process model and related assumptions

- Graph model
- Dataflow that follows control flow
- Control flow with predicate nodes

Utilized domain knowledge

- Ontology
- Business rules
Qualities of processes we learn

- **Correctness**
  - Compliance with business rules
  - Compliance with ontology
  - Compliance with execution traces
  - Well-formed products (i.e., process artifacts)

- **Reliability**

- **Robustness**
General issues in learning processes

Theoretical
- traces = sentences
- process definition = Turing machine spec

Expressiveness
- Mismatches between the process model(s) used to produce the traces and the one we use for process learning
- E.g., we cannot use a process model without iteration to learn form a trace containing an iteration

Coverage
- Actual constructs reflected in the available traces may not include some constructs in the process model used to produce them
- E.g., we cannot learn iterations from traces lack such do not include repeated activities

Robustness, real time ....

Why not utilize domain knowledge?
Process learning system (PLS)

Architecture

Components for capturing demonstration

Process Generator

Domain-specific Simulator

Process Enactor

Domain Planner

Process Miner

Facts about the process model, ontologies, business rules, and domain

{ Execution traces }

{ generated traces }

{ initial process }

{ tasks for resources to perform }

{ set of ranked plans }

{ generated traces }

{ process fragments }

{ learned process }

{ updates for process model aspects and reasoning }

{ events about sim. time, states of artifacts, resources, task execution, other environment events }
Domain Knowledge used by PLS

- Activity ontology
  - IS-A and PART-OF activity types relationships

- Resource/artifact ontology
  - Artifact well-formededness constraints

- Business rules
  - Activity precedence relationships
  - Resources utilization relationships
  - Inhibitor relationships
Artifact spec from ontology
First step in PLS

- Use knowledge about artifact structure to suggest a control flow
- Utilize well-formedness constraints
- Use an approach similar to hierarchical planning/scheduling
- Provide a more general output than planning/scheduling
**Process generator algorithm**

- **Data flow**
  - Business Rules (BRs)
  - Execution traces
  - Check Consistency
  - Set of consistency violations of BR by traces
  - Clean traces from "noise" to comply with business rules
  - Set of traces Consistent with BRs

- **Activity ontology**
  - Identify a set of possible final activities of a process
  - Set of possible final activities

- **Product artifact type structure**
  - Identify a set of possible start activities of a process
  - Set of possible start activities

- **Set of artifact elements assembled in the product artifact type structure**

- **Process synthesis task**
  - Data flow
  - Process synthesis task
  - Data
Data flow

Process synthesis

task

Data

Execution traces consistent with BRs

Activity ontology

Set of start activities

Set of final activities

Optimization criteria (goal function to determine data dependency edge weights)

Find paths in a graph of data dependencies between activities in the ontology

Set of shortest optimized paths between each final activity and each start activity in a resource dependency graph

Initial process model with control flow specs including possible branching and parallelism

• Create of a possible control flow graph from a subset of paths in a resource dependency graph per each possible final activity
• Enhance the created control flow graph to allow the execution traces be legal paths through it
• Assign and/or splits and joins for the identified control flow edges based on analysis of structures of input and output artifact types of activities
• Refine and/or splits and joins using the timing information from the execution traces

Set of start activities

Set of final activities

Optimization criteria (goal function to determine data dependency edge weights)
Execution traces consistent with BR

Activity ontology

Set of start activities

Set of final activities

Optimization criteria (goal function to determine data dependency edge weights)

Algorithm for finding paths in a graph of data dependencies between activities of the ontology

Set of shortest optimized paths between each final activity and each start activity in a data dependency graph

Creation of a possible control flow graph from a subset of paths in a data dependency graph per each possible final activity;

Enhancing the created control flow graph to allow the execution traces be legal paths through it

assigning and/or splits and joins for the identified control flow edges based on analysis of structures of input and output artifact types of activities;

Using the timing information from the execution traces to refine the assignment of and/or splits and joins

Initial process model with control flow spec including possible branching and parallelism
Future work

- Implement the suggested algorithm
- Evaluate on concrete examples
- Build the suggested architecture for integration of various learning methods
- Evaluate a number of learning methods in their usefulness for learning different aspects of processes from various problem domains
Open questions

How to learn features for robustness?

How to learn iteration?

How to learn in real-time?

How to learn general resource declarations?