Add data into business process verification

(ab)using planning tools for BPM

Sergio Tessaris joint work with Riccardo De Masellis, Chiara Di Francescomarino, Chiara Ghidini
KRDB SOS-2020 11/9/2020

online slides available on stessaris.pages.scientificnet.org/talks/sos2020
The problem with current tools
The problem with current tools

- BPM tool
- Icons designed by Wannapick.com

status ← review
status != done
status == done

BPM tool
Icons designed by Wannapick.com
- Theory is there
  - why not available?
  - what can we do to bridge the gap?
- Scalability problem?
Our research

- Lots of *implementable* frameworks and methods
- Do available tools scale up?
- How do we verify that?
Interesting tasks can be reduced to reachability; e.g.
- (proper) termination
- dead transitions
- trace repair/completion
- ...
E.g. Trace completion

- Assume model
- Given partial log
  - sequence of events + data updates
  - empty, partial, or complete
- Find complete sequence compatible with log
Classical planning

- Essentially reachability verification
- Plans/strategy as bonus
  - e.g. answer *what to do next* questions
- Strong community interested on scalability
  - International Planning Competition running since 1998
Planning and workflows

- Planning for workflows
  - *On the Disruptive Effectiveness of Automated Planning for LTLf-Based Trace Alignment* De Giacomo et al. AAAI 2017
  - *Automated Planning for Business Process Management* Marrella Journal on Data Semantics, June 2019

- Workflows for planning
Focus of this work

- How we can exploit the available planning tools
- Which tools are best suited for workflow analysis
- Evaluate scalability
Putting tools to the test

- Focus on reachability; i.e. automated planning
- Select industrial-strength tools:
  - Answer Sets Programming: Clingo
  - Classical Planning: Fast Downward
  - Model checking: nuXmv
- Build a common ground among the tools
- Select appropriate experiments
Data-aware Workflows: which one?

- Several proposed frameworks
- We selected a simple one
  - i.e. close to Classical Planning
- It’s an initial step
Data Workflow Nets (DAWNets)

- Workflow Nets
  - connected Petri Nets, with start and stop places
- Variables
  - domain
  - possibly unassigned
  - transitions assign values
- Transition guards
- based on *Soundness verification for conceptual workflow nets with data* Sidorova et al. Information Systems 2011/11
DAWNet example
DAWNets semantics

- Extends PN semantics
  - State: marking + vars
- Valid firing \( t : s \xrightarrow{} s' \)
  - \( s \): token in each input place of \( t \)
  - \( s' \): guard is satisfied
  - \( s' \): tokens from in to out places
  - \( s' \): variables updated
- Case: sequence of valid firing
Restrictions

- Bounded networks (safeness)
  - *correct* algorithms to check it
- Finite domains
Different paradigms one task

- Different tools uses different languages
  - several ad hoc encodings in literature
- Common denominator: Labelled Transition Systems
DAWNets reachability as LTS

- Labels: transition names
- States: \((M, \nu)\) marking + variable assignment
- Initial state: token in start + unassigned variables
- Transition relation: \((s, t, s')\) based on firing \(t : s \rightsquigarrow s'\)
- Goal states satisfying required properties
  - e.g. proper termination
PDDL Planners

- Reachability in LTS is a planning problem
- Actions schemata
  - pre/post conditions
- Initial conditions
- Final conditions
- Built-in frame axiom
- Operational semantics
Planners are optimised for subsets of the language
- Fast Downward
  - grounding!
- Several heuristics, some depending on PDDL subset
- E.g. no object fluents
  - only boolean predicates
- Places: constants + *active* predicate
- Transitions: actions
- Variables: unary predicates
Fluents

Causation rules to define the LTS
  - head depends on both previous and current states

\[ F \text{ if } G \text{ ifcons } H \text{ after } M. \]

\[ t;F :- t;G, \text{ not not } t;H, (t-1);M. \]

Variables, ASP style strong negation
  - grounding!

Valid states are stable models wrt the rules

Compact encoding

Native domain constraints

Inertia is not builtin
ASP planning in practice

- Language is not standardised
- Coala (based on Clingo)
  - Not optimised
- Places: unary predicate
- Transitions: actions
- Variables: unary predicates
Model checking

- Tools are based on TL over infinite traces (LTL)
  - looping on final states
- Variables over arbitrary domains or booleans
- TS defined using formulae over current and previous states
- Native constraints over domain
- Inertia is not builtin
  - no NMR to help with that
Model checking in practice

- nuXmv
- Places: boolean variables
- Transitions: variable over transition names
- Variables
Encodings

- More details in my early talk
- Leveraging trace equivalence
  - formally proven for each encoding
- Transferable models
PDDL encoding (Domain)

(:constants
  p2 p3 p1 p6 p7 p4 p5 p8 p9 start end - place
  high s low w mid - active_domain
)

(:predicates
  ;; Places
  (p_enabled ?p - place )
  (p_terminal ?p - place )
  ;; Variables
  (request ?v - active_domain )
  ;; Domains
  (t4_request_domain ?v - active_domain)
)
(:action t4
  :parameters ( ?request - active_domain )
  :precondition
    (and (p_enabled p2) (t4_request_domain ?request))
  :effect
    (and
      (p_enabled p4)
      (not (p_enabled p2))
      (forall (?v - active_domain) (not (request ?v)))
      (request ?request)))
PDDL encoding (Problem)

(:init
  (p_enabled start)
  (p_terminal end)
  (t4_request_domain low)
  (t4_request_domain mid)
  (t4_request_domain high)
)

(:goal
  (and
  (p_enabled end)
  (forall (?p - place)
    (or (p_terminal ?p) (not (p_enabled ?p)))))))
Which experiments?

- Difficult to design general reachability experiments
- Focus on *Trace Completion*
- Several parameters via traces:
  - completeness degree
  - compliance
  - size
Trace Completion as Reachability
Synthetic models

- Synthetic base model
- Combination of copies of the base model
- 8 different traces per model (size 10-50+)
  - empty, 25%, 50%, 75%
  - also not compliant (4)
Base model

M1

M2

M3

M4

M5
Models
Real life model

- BPI Challenge 2011 logs
  - *Real life log of a Dutch academic hospital*
- Model discovered using ProM Data-flow Discovery plugin
  - 355 transitions, 61 places, 710 edges, and 4 variables
- 9 random traces (size 3-500+)
  - empty, 25%, 50%, 75%
  - all compliant
- Also tested without data
Discovered model
It’s not reproducible if it only runs on your laptop

Jon Zelner

- Focus on reproducibility
- Leveraging Docker, and Kubernetes cluster
- More details on Use Containers for your Experiments!
Have we a winner?

- Simple answer: no
Have we a winner?

- Simple answer: no
- Actually, the picture is more complex... let's look at some pictures
- Zooming into FD/clingo
Real life experiments

![Graph showing the performance of solvers (clingo and nuXmv) across different completeness levels. The graph indicates the number of timeouts and out of memory conditions as the completeness percentage increases.](image-url)
What if we get rid of data?
What if we get rid of data?
Ground or not to ground

- Grounding can be expensive!
- For *big* problems symbolic verification can be beneficial
- Room for hybrid methods?
Optimising encoding

- We didn’t optimise encoding and heuristics
- Need collaboration with tool developers
  - E.g. direct encoding in SAS for FD
Conclusions

- Automated planning tools are effective
- Adding data makes the difference
- Hybrid systems can be a solution
- Now we can move to more complex languages
Questions?
References

- Solving reachability problems on data-aware workflows
  - code available
- Use Containers for your Experiments!
- Verification of workflow nets with data