ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Towards the Generation of the "Perfect" Log Using Abductive Logic Programming

F. Chesani, C. Di Francescomarino, C. Ghidini, D. Loreti, F. M. Maggi, P. Mello, M. Montali, V. Skydanienko, and S. Tessaris

CILC 2019
34th Italian Conference on Computational Logic
19-21 June 2019, Trieste, Italy

The Business Process Management research field

The research field is focused on many different aspects:

- **Process modeling** languages and semantics
  - Procedural approaches, such as BPMN
  - Declarative approaches, such as Declare

- **Process Mining**
  - Formal properties verification (of process models)
  - Compliance and conformance verification (of logs versus models)
  - Process discovery (mining in a more "classical" terminology)

- … (many others)

About the Logs… (1/2)

Logs play a fundamental role

- Process discovery algorithms -> their evaluation is possible only starting from logs
- Predictive process monitoring -> same as above
- Repairing
- Process Conformance analysis (run-time, and post-mortem)
- Process Quality Analysis, Assurance, Auditing
- Real application cases: process models are not given a–priori, but learned by logs of observed process instances

The official format XES has been defined within the BPM community

- A log is a collection of information about process instances
- All the data relative to a single process instance is named **trace,** with its **trace id**
- A trace is a collection of (happening of) **events**, where each event captures the execution of an activity
- Minimum requirements for each event: **Event Description**, **Timestamp**, **Start/End** (or both) of an activity

About the Logs… (2/2)

Real logs are intrinsically “positive”

- Real logs, when available, represent the effective running of some business
- The business owner will always qualify them as “correct”
- Even in case “negative” traces are in the log, they are very few…

What about data?

- Data in real log is very rare
    - Rarely, there is too much data (big data approach: “let us log everything”)… process discovery approaches are confused!
    - More often, some data field are completely missing, while other data filed are only partially recorded in the log…

Real logs are scarce!

The Quest for the Perfect Log

Many researchers have turned towards Synthetic Log Generators

- They takes as input a process model (procedural or declarative, open or closed)
- They provide as output a log, that exhibits the desired features

Which characteristics of the log?

- It mainly depend on the intended use of the log
- In case of process discovery, also the discovery algorithm can be taken into account when generating the log
- However, some desirable features:
    - Positive, and also negative traces should be available
    - User-definable balance between #positive vs. #negatives
    - Flows and execution paths full/partial coverage
    - Data-domain coverage and distribution
    - Time-domain coverage and distribution

Synthetic Logs – Existing approaches…
(a very partial list)

- Medeiros, A.K.A.D., De Medeiros, A.K.A., Gu ̈nther, C.W.: Process mining: Using cpn tools to create test logs for mining algorithms. Procs. of the 6th works. on practical use of coloured petri nets and the cpn tools pp. 177–190 (2005)
- ynn, M.T., Dumas, M., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Business process simulation for operational decision support. In: BPM Workshops, BPM 2007. LNCS, vol. 4928, pp. 66–77. Springer (2007)
- Burattin, A., Sperduti, A.: PLG: A framework for the generation of business pro- cess models and their execution logs. In: BPM 2010 Workshops. LNBIP, vol. 66, pp. 214–219. Springer (2010)
- van Hee, K.M., Liu, Z.: Generating benchmarks by random stepwise refinement of petri nets. In: PETRI NETS 2010. CEUR Workshop Proceedings, vol. 827, pp. 403– 417. CEUR-WS.org (2010)
- Westergaard, M., Slaats, T.: CPN tools 4: A process modeling tool combining declarative and imperative paradigms. In: BPM Demo sessions 2013, Procs. CEUR Procs., vol. 1021. CEUR-WS.org (2013)
- Stocker, T., Accorsi, R.: Secsy: A security-oriented tool for synthesizing process event logs. In: Procs. of the BPM Demo Sessions 2014. CEUR Procs., vol. 1295, p. 71. CEUR-WS.org (2014)
- Van den Broucke, S.: Advances in Process Mining: Artificial negative events and othertechniques. Ph.D. thesis, Katholieke Universiteit Leuven, Belgium (2014)
- Di Ciccio, C., Bernardi, M.L., Cimitile, M., Maggi, F.M.: Generating event logs through the simulation of declare models. In: EOMAS 2015, Held at CAiSE 2015. LNBIP, vol. 231, pp. 20–36. Springer (2015)
- Ackermann, L., Scho ̈nig, S., Jablonski, S.: Simulation of multi-perspective declarative process models. In: BPM 2016 Works. LNBIP, vol. 281, pp. 61–73 (2016)

Previously, on these screens… (CILC2016, Milan)

We investigated the problem of determining the conformance of a log/a trace vs. a process model

- Input 1: a process model in YAWL, a procedural closed language (for process modeling)
- Input 2: a log
- Output: conformance of the observed log/trace w.r.t. the process model

Key point: the approach supported incompleteness of the log / of the traces / of the single events

How? By means of Abduction

- When some data is missing, let us hypothesize (abduce) the missing information
- The abductive answer Delta indicates the set of needed assumptions

F. Chesani, P. Mello, R. De Masellis, C. Di Francescomarino, C. Ghidini, M. Montali, S. Tessaris:
Compliance in Business Processes with Incomplete Information and Time Constraints: a General Framework based on Abductive Reasoning.
Fundam. Inform. 161(1-2): 75-111 (2018)

Abduction and SCIFF Framework

An Abductive Logic Program [Kakas et al., 1993] is a trile $\langle \mathcal{KB}, \mathcal{A}, \mathcal{IC} \rangle$ where:

(i) $\mathcal{KB}$ is a (static) knowledge base (a Logic Program);

(ii) $\mathcal{A}$ is a special set of predicates, called *abducibles*.

(iii) $\mathcal{IC}$ is a set of integrity constraint.

Given a goal $\mathcal{G}$, abductive reasoning looks for a set of literals $\Delta \subseteq \mathcal{A}$ such that the goal is entailed by the program $\mathcal{KB} \cup \Delta$, and the set of integrity constraints $\mathcal{IC}$ is entailed too. The set Δ is referred to as an *abductive explanation*.

Antonis C. Kakas, Robert A. Kowalski, Francesca Toni:
Abductive Logic Programming. J. Log. Comput. 2(6): 719-770 (1992)

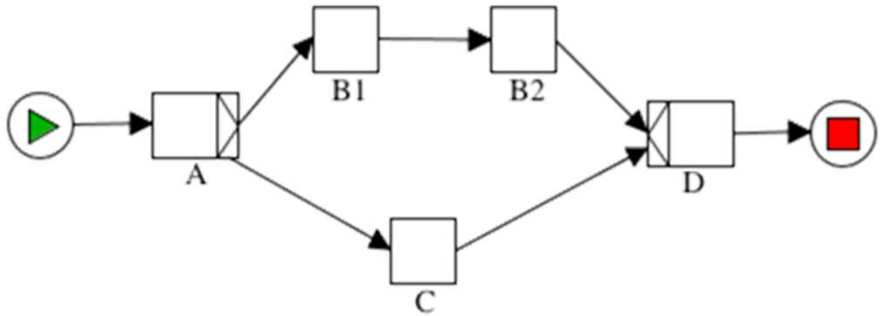
M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, P. Torroni:
Verifiable agent interaction in abductive logic programming: The SCIFF framework. ACM Trans. Comput. Log. 9(4): 29:1-29:43 (2008)

SCIFF is a Framework for ALP, plus:

- Happened Events
  HAP(Desc, T)
- Expectations:
  E(Desc, T)
- Prohibitions:
  EN(Desc, T)
- General abducibles:
  ABD(Desc, T)

- ICs are forward rules, containing variables
- Variables can be constrained (CLP)

Example…



Let us suppose that activity **B2 sometimes is not observed**… hence we model the sequence B1-B2 as:

H(b1, Tb1) ---> **E**(b2, Tb2) /\ Tb2>Tb1
\/
ABD(b2, Tb2) /\ Tb2>Tb1.

Suppose we observe a trace:

(a,2)
(b1, 5)
(d, 10)

The abductive proof procedure would say that the trace is *compliant* if we can hypothesize (b2, T2), 5<T2<10

{ (b2, T2), 5 < T2 < 10 } is the *abductive answer.*

Starting from that previous work…

Question: what if we provide in input an **empty trace**?

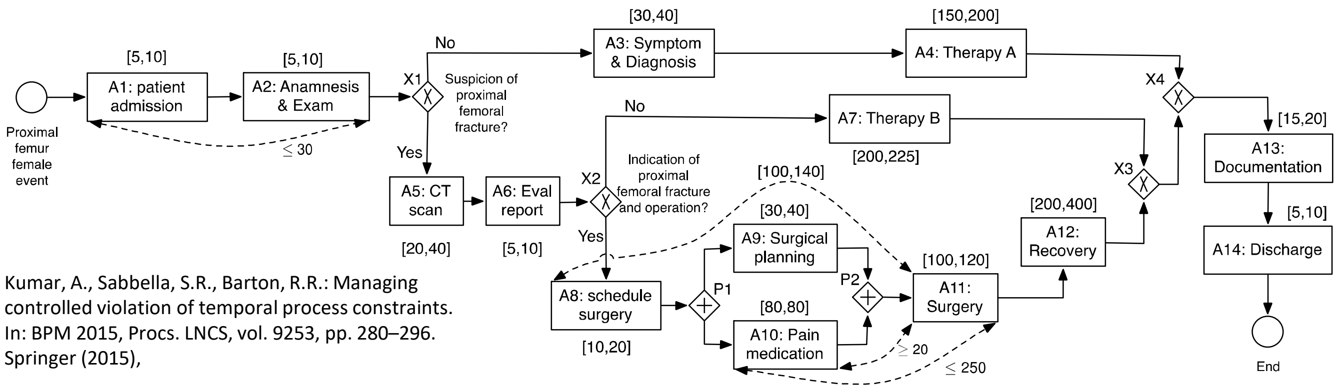
Answer: the abductive procedure hypothesize the happening of ALL the missing information, so as to have a trace that is compliant with the process model…

… it generates a complete trace!!!

IDEA: let us use this same approach to generate synthetic traces/logs

The case of procedural, closed languages

- First attempt was presented in the AI4BPM Workshop at the BPM conference 2017
- Process model specified through a procedural, closed language
- Limited to the generation of positive traces only
- No data, but temporal constraints on activity durations, and between activities



Kumar, A., Sabbella, S.R., Barton, R.R.: Managing controlled violation of temporal process constraints. In: BPM 2015, Procs. LNCS, vol. 9253, pp. 280–296. Springer (2015),

```
ABD(start,Tstart),
ABD(a1_start, Ta1_start),
ABD(a1_end, Ta1_end),
ABD(a2_start, Ta2_start),
ABD(a2_end, Ta2_end),
ABD(a3_start,Ta3_start),
ABD(a3_end, Ta3_end),
ABD(a4_start, Ta4_start),
ABD(a4_end, Ta4_end),
ABD(a13_start, Ta13_start),
ABD(a13_end, Ta13_end),
ABD(a14_start, Ta14_start),
ABD(a14_end, Ta14_end),
ABD(stop, Tstop )
```

F. Chesani, A. Ciampolini, D. Loreti, P. Mello: Abduction for Generating Synthetic Traces. Business Process Management Workshops 2017: 151-159

Today’s work

- Process model specified through a declarative, open language: DECLARE
- Generation of positive and negative traces
- Data!!!

TEMPLATE	FORMALIZATION	NOTATION	DESCRIPTION
existence(A)	$\Diamond A$	1..* A	A occurs at least once
init(A)	A	init A	A is the first event to occur
resp. existence(A,B)	$\Diamond A \rightarrow \Diamond B$	A B	If A occurs, B must occur as well
response(A,B)	$\Box(A \rightarrow \Diamond B)$	A B	If A occurs, B must eventually follow
precedence(A,B)	$\neg B \, \mathcal{W} \, A$	A B	B can occur only if A has occurred before
chain response(A,B)	$\Box(A \rightarrow \bigcirc B)$	A B	If A occurs, B must occur next
alt. succession(A,B)	$(\neg B \, \mathcal{W} \, A) \wedge$ $\Box(B \rightarrow \bigcirc(\neg B \, \mathcal{W} \, A)) \wedge$ $\Box(A \rightarrow \bigcirc(\neg A \, \mathcal{U} \, B))$	A B	A and B occur if and only if B follows A and they alternate each other
chain succession(A,B)	$\Box(A \longleftrightarrow \bigcirc B)$	A B	A occurs if and only if B immediately follows

DECLARE Notation (partial list)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Generating positive traces…

1. **Translate** the DECLARE model into ALP
   No need of proving the correctness of the translation because… DECLARE is a graphical language, we provided semantics to its symbols by means of ALP Integrity Constraints

2. **Generate an abductive answer**
   The answer will contain variables, with associated domains

3. **Ground the variables** by asking the underlying CLP solver one or more solutions

Generating positive traces: Step 1

Two set of ALP Integrity Constraints:

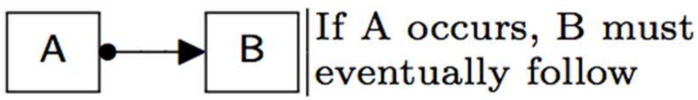
1. First set captures the generation of activities in an **open world**
   For each activity X envisaged in the model:

    true ---> true
    \/
    ABD(X, Tx).

    ABD(X, Tx) ---> true
    \/
    ABD(X, Tx1) /\ Tx1 > Tx.

Does it terminate ???????

2. Second set captures DECLARE constraints:

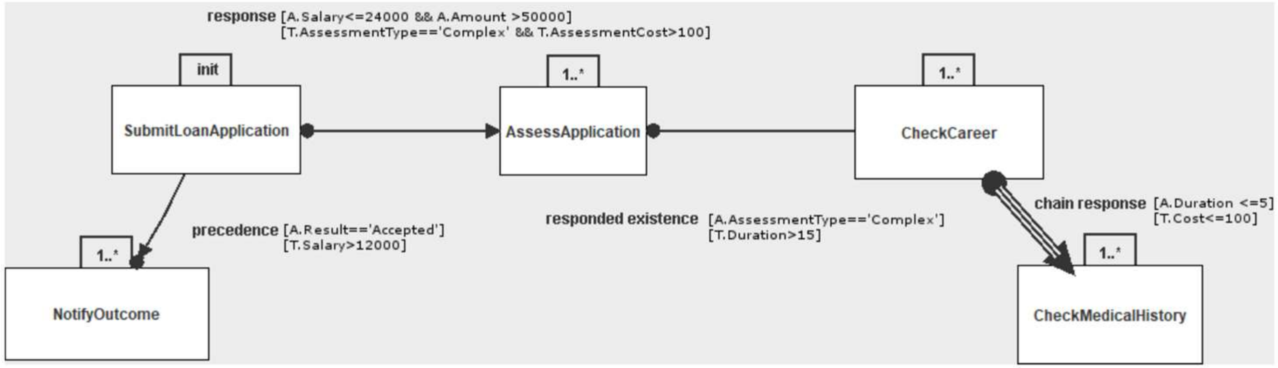


ABD(a, Ta) ---> ABD (b, Tb) /\ Tb > Ta.

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Generating positive traces: Step 2 & 3

Step 2 is achieved directly by the ALP Proof Procedure: our choice is the SCIFF Framework.



$$\begin{aligned}
\tau = \{&abd(submitLoanApplication(Salary,_A mount), T1),\\
&abd(assessApplication(1, AssessmentCost), T2),\\
&abd(checkCareer(Coverage), T3),\\
&abd(checkMedicalHistory(Cost), T4),\\
&abd(event(notifyOutcome(0), T5),\\
&Salary : 12001 \;..\; 24000,\\
&Amount : 50001 \;..\; 299999,\\
&T1 : 1 \;..\; 8,\\
&AssessmentCost : 101..199,\\
&T2 : 2 \;..\; 9,\\
&Coverage : 16 \;..\; 29,\\
&T3 : 2 \;..\; 9,\\
&Cost : 11 \;..\; 199,\\
&T4 : 2 \;..\; 9,\\
&T5 : 2 \;..\; 9\}
\end{aligned}$$

Step 3 is achieved by asking the constraint solver to label the instances.

Generating negative traces… how?

1. Translate the DECLARE model into ALP
2. NEGATE the model
3. Generate an abductive answer
4. Ground the variables by asking the underlying CLP solver one or more solutions

Any trace generated in this way, it will violate the initial model…

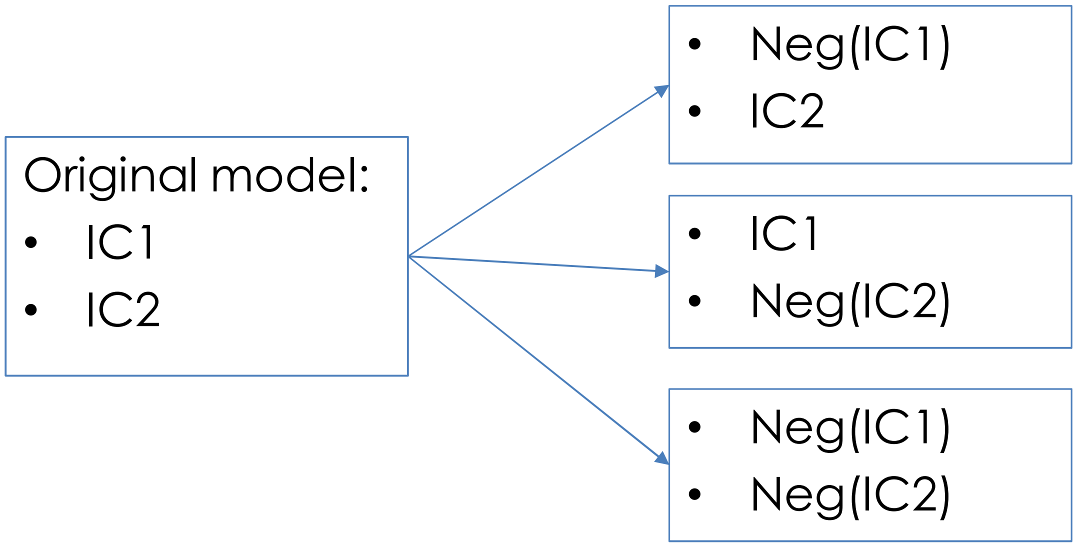
“Negating” a model … What does it mean?

Few observation:

a) A ALP model is a conjunction of Integrity Constraints.

b) If a trace violates a model, it means that it violates *one or more* Integrity Constraints.

(#1) COMBINATORY EXPLOSION of negated models, given by the negation of the powerset of the Integrity Constraints of the original model…



Remark: Not all the resulting models will be consistent, some of them will never lead to the generation of a trace

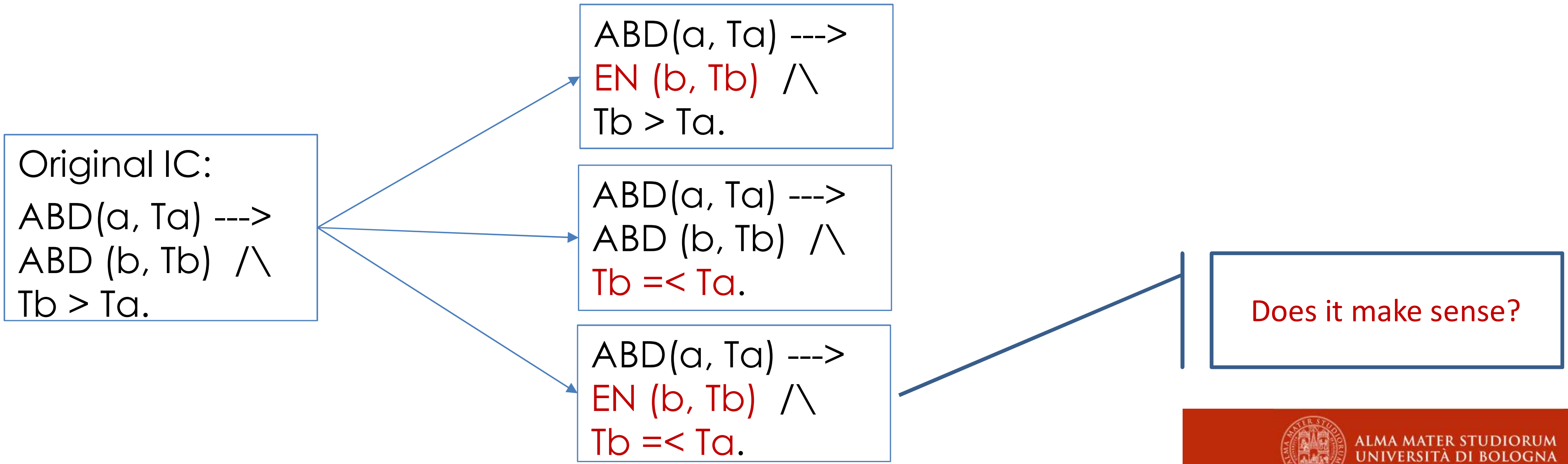
“Negating” an IC … What does it mean?

Few observation:

a) An IC is an implication: it is violated when the premises are true and the consequences are false.

b) Consequences, in the most simple case, are a conjuction of predicates.

(#2) COMBINATORY EXPLOSION of negated IC, given by the negation of the powerset of the conjuncts in the original IC



Last issue… the grounding of data

- Grounding is achieved through the labeling procedure.
- Through fail and backtracking, it is possible to get ALL the possible groundings.
- But it does not make sense! Very boring logs…

Current (naïve) solution:

1. Ask the user for a number N
2. For each variable X:
    1. Get the max and the min value of the domain, and compute delta = (max-min)/N
    2. For i=0 to N iterate:
        1. Get a grounding through the labelling procedure
        2. Impose a fail
        3. Add the constraint X > i*delta

The objective is to cover the data domains with some distribution…

In any case, (#3) Exponential Explosion of the grounded traces

Upperbound: N^(NumberOfVariables)

Work done so far

- Automatic translation from a DECLARE or a YAWL-based process model into SCIFF
    - Supports procedural and declarative modelling languages
    - Supports open and closed modelling languages
- Automatic generation of positive and negative traces
- Grounding using a sort-of uniform distribution

- Initial tentatives of learning the log *again*…

Current work:

Does the generated log make sense?

Brief recap:

- We start from a model of the process
- We generate a log of positive and negative traces

Let us learn again the model, and see what happens

Which is the perfect log?

- The one that covers all the paths?
- The one that covers all the data values?

How is the space of traces shaped?

- How many positive traces?
- How many negative traces?

Thanks for your time!!!

Questions?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA