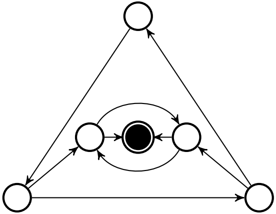
unibz

Monitoring Business Metaconstraints Based on LTL & LDL for Finite Traces



Marco Montali

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano

Joint work with: G. De Giacomo, R. De Masellis, M. Grasso, F.M. Maggi

BPM 2014

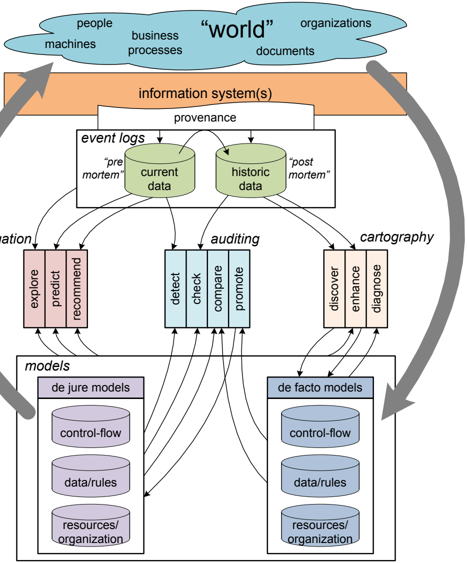
Process Mining

Process Mining

Classicaly applied to **post-mortem** data.

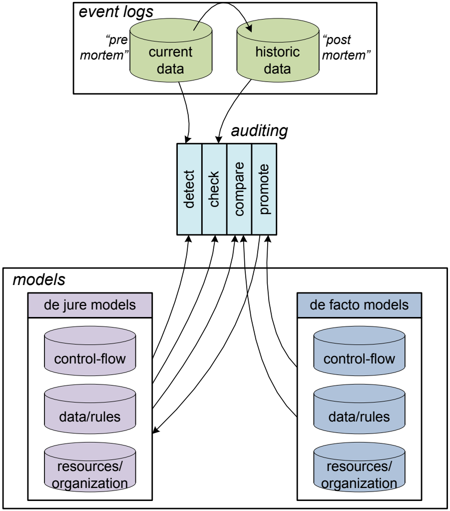
Operational Decision Support

Extension of classical process mining to current, **live** data.



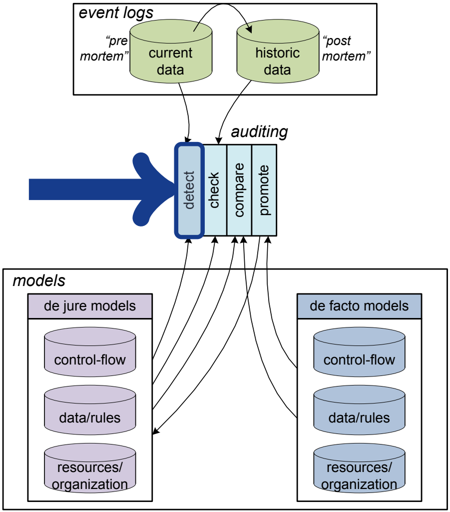
Detecting Deviations

Auditing: find deviations between observed and expected behaviors.



Detecting Deviations

Auditing: find deviations between observed and expected behaviors.



Our setting:

Model

Declarative business constraints.

- E.g., **Declare**.

Monitoring

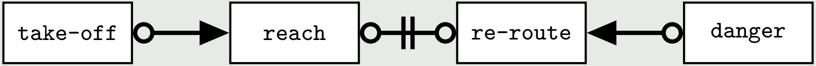
- Online, evolving observations.
- Prompt deviation detection.

On Promptness

unibz

Flight routes (thanks Claudio!)

- When the airplane takes off, it must eventually reach the destination airport.
- When the airplane is re-routed, it cannot reach the destination airport anymore.
- If a dangerous situation is detected at the destination, airplane must be re-routed.



Question

Consider trace:

$$\texttt{take-off} \longrightarrow \texttt{danger}$$

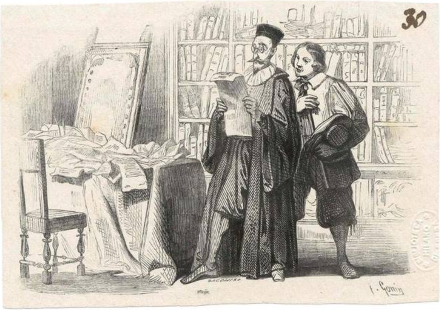
Is there any deviation?

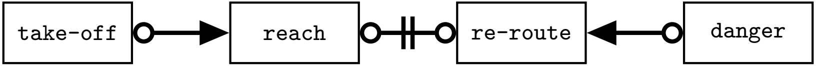
Boring Answer: **Apparently Not**

unibz

Reactive Monitor

- Checks the partial trace observed so far.
- Suspends the judgment if no conclusive answer can be given.





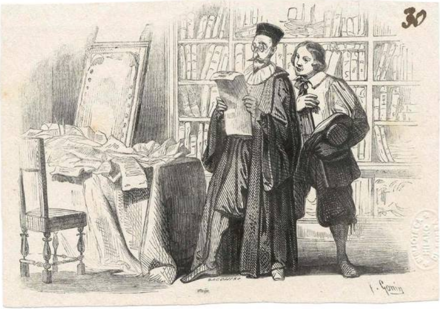
take-off ⟶ danger

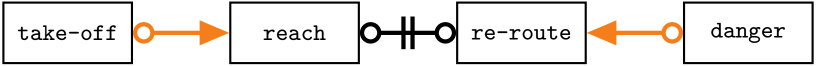
Boring Answer: **Apparently Not**

unibz

Reactive Monitor

- Checks the partial trace observed so far.
- Suspends the judgment if no conclusive answer can be given.



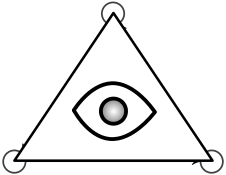


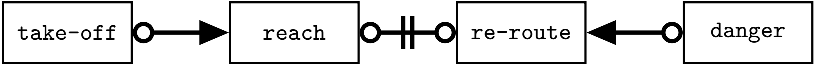
take-off ⟶ danger

Prophetic Answer: **YES**

Proactive Monitor

- Checks the partial trace observed so far.
- Looks into the future(s).





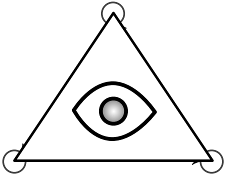
`take-off` ⟶ `danger`

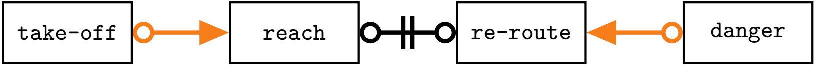
Prophetic Answer: **YES**

unibz

Proactive Monitor

- Checks the partial trace observed so far.
- Looks into the future(s).



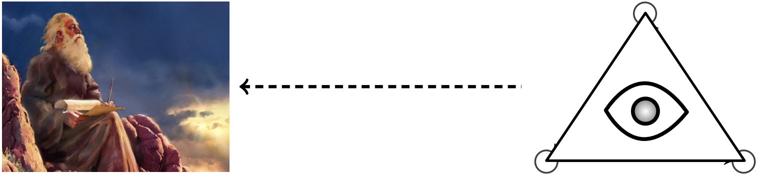


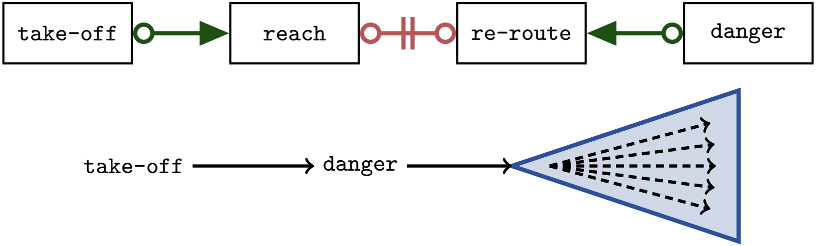
take-off ⟶ danger

Prophetic Answer: **YES**

Proactive Monitor

- Checks the partial trace observed so far.
- Looks into the future(s).

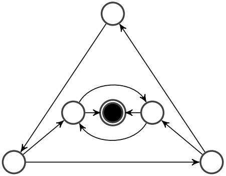




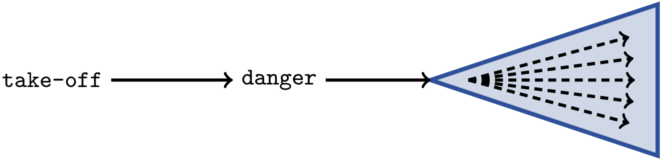
Prophetic Answer: **YES**

Proactive Monitor

- Checks the partial trace observed so far.
- Looks into the future(s).



$$\Box(\texttt{take-off} \rightarrow \Diamond\texttt{reach}) \wedge \neg(\Diamond(\texttt{reach}) \wedge \Diamond(\texttt{re-route})) \wedge \Box(\texttt{danger} \rightarrow \Diamond\texttt{re-route})$$



Logics on Finite Traces

unibz

Goal

Reasoning on finite partial traces and their finite suffixes.

Typical Solution: LTL_f

Adopt LTL on finite traces and corresponding techniques based on Finite-State Automata.[a]

[a] Not Büchi automata!

Logics on Finite Traces

unibz

Goal

Reasoning on finite partial traces and their finite suffixes.

Typical Solution: LTL_f

Adopt LTL on finite traces and corresponding techniques based on Finite-State Automata.[a]

[a] Not Büchi automata!

Huge difference, often neglected, between LTL on finite and infinite traces! See [AAAI2014]

Problem #1: Monitoring

Proactive monitoring requires to refine the standard LTL_f semantics.

RV-LTL

Given an LTL formula φ:

- $[\varphi]_{RV} = \mathit{true} \rightsquigarrow$ OK;
- $[\varphi]_{RV} = \mathit{false} \rightsquigarrow$ BAD;
- $[\varphi]_{RV} = \mathit{temp\_true} \rightsquigarrow$ OK now, could become BAD in the future;
- $[\varphi]_{RV} = \mathit{temp\_false} \rightsquigarrow$ BAD now, could become OK in the future.

Problem #1: Monitoring

Proactive monitoring requires to refine the standard LTL_f semantics.

RV-LTL

Given an LTL formula φ:

- $[\varphi]_{RV} = true \rightsquigarrow$ OK;
- $[\varphi]_{RV} = false \rightsquigarrow$ BAD;
- $[\varphi]_{RV} = temp\_true \rightsquigarrow$ OK now, could become BAD in the future;
- $[\varphi]_{RV} = temp\_false \rightsquigarrow$ BAD now, could become OK in the future.

However...

- Typically studied on infinite traces: detour to Büchi automata.
- Only ad-hoc techniques on finite traces [BPM2011].

Problem #2: Contextual Business Constraints

Need for monitoring constraints only when specific circumstances hold.

- Compensation constraints.
- Contrary-do-duty expectations.
- ...

Problem #2: Contextual Business Constraints

Need for monitoring constraints only when specific circumstances hold.

- Compensation constraints.
- Contrary-do-duty expectations.
- ...

However...

- Cannot be systematically captured at the level of constraint specification.

Suitability of the Constraint Specification Language

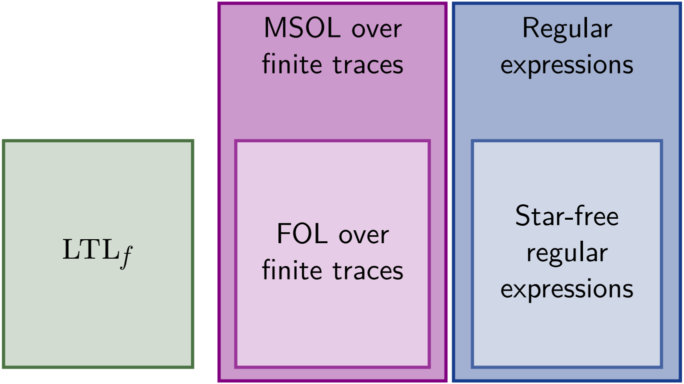
unibz

LTL_f

FOL over finite traces

Star-free regular expressions

Suitability of the Constraint Specification Language



Suitability of the Constraint Specification Language

unibz

LTL_f

MSOL over finite traces

FOL over finite traces

Regular expressions

Star-free regular expressions

PSPACE complexity

Nondet. finite-state automata (NFA)

- LTL_f: declarative, but lacking expressiveness.
- Regular expressions: rich formalism, but low-level.

(t)ake-off → (r)each $\rightsquigarrow ((\mathtt{r}|\mathit{other})^*(\mathtt{t}(\mathtt{t}|\mathit{other})^*\mathtt{r})(\mathtt{r}|\mathit{other})^*)^*$

Suitability of the Constraint Specification Language

unibz

LDL_f **Linear Dynamic Logic over finite traces**	MSOL over finite traces	Regular expressions	PSPACE complexity
LTL_f	FOL over finite traces	Star-free regular expressions	Nondet. finite-state automata (NFA)

- LTL_f: declarative, but lacking expressiveness.
- Regular expressions: rich formalism, but low-level.

  `(t)ake-off` ○—▶ `(r)each` $\rightsquigarrow$ $((\mathtt{r}|\mathit{other})^*(\mathtt{t}(\mathtt{t}|\mathit{other})^*\mathtt{r})(\mathtt{r}|\mathit{other})^*)^*$
- LDL_f: combines the best of the two!

The Logic LDL_f [De Giacomo&Vardi,IJCAI13]

Merges LTL_f with regular expressions, through the syntax of Propositional Dynamic Logic (PDL):

$$\begin{array}{rcl} \varphi & ::= & \phi \mid tt \mid ff \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\rho\rangle\varphi \mid [\rho]\varphi \\ \rho & ::= & \phi \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1;\rho_2 \mid \rho^* \end{array}$$

φ: LTL_f part; ρ: regular expression part.
They mutually refer to each other:

- $\langle\rho\rangle\varphi$ states that, from the current step in the trace, *there is* an execution satisfying ρ such that its last step satisfies φ.
- $[\rho]\varphi$ states that, from the current step in the trace, *all* execution satisfying ρ are such that their last step satisfies φ.
- $\varphi?$ checks whether φ is true in the current step and, if so, continues to evaluate the remaining execution.

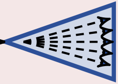
Of special interest is $end = [true?]ff$, to check whether the trace has been completed (the remaining trace is the empty one).

Runtime LDL_f Monitors

Check partial trace $\pi = e_1, \ldots, e_n$ against formula φ.

From ad-hoc techniques ...

$e_1 \rightarrow \cdots \rightarrow e_n \rightarrow$

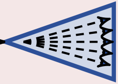


$$\models [\varphi]_{RV} = \begin{cases} \mathit{temp\_true} \\ \mathit{temp\_false} \\ \mathit{true} \\ \mathit{false} \end{cases}$$

Runtime LDL_f Monitors

Check partial trace $\pi = e_1, \ldots, e_n$ against formula φ.

From ad-hoc techniques ...

$e_1 \rightarrow \cdots \rightarrow e_n \rightarrow$  $\models [\varphi]_{RV} = \begin{cases} \mathit{temp\_true} \\ \mathit{temp\_false} \\ \mathit{true} \\ \mathit{false} \end{cases}$

... To standard techniques

$$e_1 \rightarrow \cdots \rightarrow e_n \models \begin{cases} \varphi_{\mathit{temp\_true}} \\ \varphi_{\mathit{temp\_false}} \\ \varphi_{\mathit{true}} \\ \varphi_{\mathit{false}} \end{cases}$$

How is This Magic Possible?

Starting point: LDL_f formula φ and its RV semantics.

How is This Magic Possible?

unibz

Starting point: LDL_f formula φ and its RV semantics.

1. Good and bad prefixes

- $\mathcal{L}_{poss\_good}(\varphi) = \{\pi \mid \exists \pi'.\pi\pi' \in \mathcal{L}(\varphi)\}$
- $\mathcal{L}_{nec\_good}(\varphi) = \{\pi \mid \forall \pi'.\pi\pi' \in \mathcal{L}(\varphi)\}$
- $\mathcal{L}_{nec\_bad}(\varphi) = \mathcal{L}_{nec\_good}(\neg\varphi) = \{\pi \mid \forall \pi'.\pi\pi' \notin \mathcal{L}(\varphi)\}$

How is This Magic Possible?

unibz

Starting point: LDL_f formula φ and its RV semantics.

1. Good and bad prefixes

- $\mathcal{L}_{poss\_good}(\varphi) = \{\pi \mid \exists \pi'.\pi\pi' \in \mathcal{L}(\varphi)\}$
- $\mathcal{L}_{nec\_good}(\varphi) = \{\pi \mid \forall \pi'.\pi\pi' \in \mathcal{L}(\varphi)\}$
- $\mathcal{L}_{nec\_bad}(\varphi) = \mathcal{L}_{nec\_good}(\neg\varphi) = \{\pi \mid \forall \pi'.\pi\pi' \notin \mathcal{L}(\varphi)\}$

2. RV-LTL values as prefixes

- $\pi \models [\varphi]_{RV} = true$ iff $\pi \in \mathcal{L}_{nec\_good}(\varphi)$;
- $\pi \models [\varphi]_{RV} = false$ iff $\pi \in \mathcal{L}_{nec\_bad}(\varphi)$;
- $\pi \models [\varphi]_{RV} = temp\_true$ iff $\pi \in \mathcal{L}(\varphi) \setminus \mathcal{L}_{nec\_good}(\varphi)$;
- $\pi \models [\varphi]_{RV} = temp\_false$ iff $\pi \in \mathcal{L}(\neg\varphi) \setminus \mathcal{L}_{nec\_bad}(\varphi)$.

How is This Black Magic Possible?

unibz

3. Prefixes as regular expressions

Every NFA can be expressed as a regular expression.

$\rightsquigarrow$ We can build regular expression pref_{φ} s.t. $\mathcal{L}(\mathsf{pref}_{\varphi}) = \mathcal{L}_{poss\_good}(\varphi)$.

How is This Black Magic Possible?

3. Prefixes as regular expressions

Every NFA can be expressed as a regular expression.
$\rightsquigarrow$ We can build regular expression pref_φ s.t. $\mathcal{L}(\mathsf{pref}_\varphi) = \mathcal{L}_{poss\_good}(\varphi)$.

4. Regular expressions can be immersed into LDL_f

Hence: $\pi \in \mathcal{L}_{poss\_good}(\varphi)$ iff $\pi \models \langle \mathsf{pref}_\varphi \rangle end$
$\pi \in \mathcal{L}_{nec\_good}(\varphi)$ iff $\pi \models \langle \mathsf{pref}_\varphi \rangle end \land \neg \langle \mathsf{pref}_{\neg\varphi} \rangle end$

How is This Black Magic Possible?

3. Prefixes as regular expressions

Every NFA can be expressed as a regular expression.

$\rightsquigarrow$ We can build regular expression pref_φ s.t. $\mathcal{L}(\mathsf{pref}_\varphi) = \mathcal{L}_{poss\_good}(\varphi)$.

4. Regular expressions can be immersed into LDL_f

$$\text{Hence: } \begin{array}{ll} \pi \in \mathcal{L}_{poss\_good}(\varphi) & \text{iff } \pi \models \langle \mathsf{pref}_\varphi \rangle end \\ \pi \in \mathcal{L}_{nec\_good}(\varphi) & \text{iff } \pi \models \langle \mathsf{pref}_\varphi \rangle end \wedge \neg \langle \mathsf{pref}_{\neg\varphi} \rangle end \end{array}$$

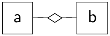
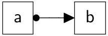
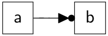
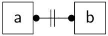
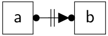
5. RV-LTL can be immersed into LDL_f!

- $\pi \models [\varphi]_{RV} = true$ iff $\langle \mathsf{pref}_\varphi \rangle end \wedge \neg \langle \mathsf{pref}_{\neg\varphi} \rangle end$;
- $\pi \models [\varphi]_{RV} = false$ iff $\langle \mathsf{pref}_{\neg\varphi} \rangle end \wedge \neg \langle \mathsf{pref}_{\varphi} \rangle end$;
- $\pi \models [\varphi]_{RV} = temp\_true$ iff $\pi \models \varphi \wedge \langle \mathsf{pref}_{\neg\varphi} \rangle end$;
- $\pi \models [\varphi]_{RV} = temp\_false$ iff $\pi \models \neg\varphi \wedge \langle \mathsf{pref}_{\varphi} \rangle end$.

Ending point: 4 LDL_f monitor formulae under standard semantics.

Monitoring DECLARE with LDL$_f$

Step 1. Good prefixes of DECLARE patterns.

	NAME	NOTATION	pref	POSSIBLE RV STATES
EXISTENCE	**Existence**		$(a+o)^*$	*temp_false*, *true*
	Absence 2		$o^* + (o^*; a; o^*)$	*temp_true*, *false*
CHOICE	**Choice**		$(a+b+o)^*$	*temp_false*, *true*
	Exclusive Choice		$(a+o)^* + (b+o)^*$	*temp_false*, *temp_true*, *false*
RELATION	**Resp. existence**		$(a+b+o)^*$	*temp_true*, *temp_false*, *true*
	Response		$(a+b+o)^*$	*temp_true*, *temp_false*
	Precedence		$o^*; (a; (a+b+o)^*)^*$	*temp_true*, *true*, *false*
NEGATION	**Not Coexistence**		$(a+o)^* + (b+o)^*$	*temp_true*, *false*
	Neg. Succession		$(b+o)^*; (a+o)^*$	*temp_true*, *false*

Monitoring DECLARE with LDL_f

Step 1. Good prefixes of DECLARE patterns.

	NAME	NOTATION	pref	POSSIBLE RV STATES
EXISTENCE	**Existence**	$\overset{1..*}{\boxed{\text{a}}}$	$(a+o)^*$	*temp_false*, *true*
	Absence 2	$\overset{0..1}{\boxed{\text{a}}}$	$o^* + (o^*; a; o^*)$	*temp_true*, *false*
CHOICE	**Choice**	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(a+b+o)^*$	*temp_false*, *true*
	Exclusive Choice	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(a+o)^* + (b+o)^*$	*temp_false*, *temp_true*, *false*
RELATION	**Resp. existence**	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(a+b+o)^*$	*temp_true*, *temp_false*, *true*
	Response	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(a+b+o)^*$	*temp_true*, *temp_false*
	Precedence	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$o^*; (a; (a+b+o)^*)^*$	*temp_true*, *true*, *false*
NEGATION	**Not Coexistence**	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(a+o)^* + (b+o)^*$	*temp_true*, *false*
	Neg. Succession	$\boxed{\text{a}}$ $\boxed{\text{b}}$	$(b+o)^*; (a+o)^*$	*temp_true*, *false*

Step 2. Generate LDL_f monitors.

- Local monitors: 1 formula for possible RV constraint state.
- Global monitors: 4 RV formulae for the conjunction of all constraints.

One Step Beyond: Metaconstraints

unibz

LDL_f monitors are simply LDL_f formulae.
They can be combined into more complex LDL_f formulae!

- E.g., expressing conditional/contextual monitoring.

Business metaconstraint

An LDL_f formula of the form $\boxed{\Phi_{pre} \rightarrow \Psi_{exp}}$

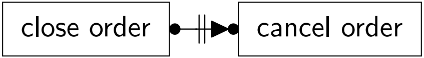
- Φ_{pre} combines membership assertions of business constraints to their RV truth values.
- Ψ_{exp} combines business constraints to be checked when Φ_{pre} holds.

LDL_f metaconstraint monitors

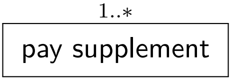
- $\Phi_{pre} \rightarrow \Psi_{exp}$ is a standard LDL_f formula.
- Hence, just reapply our technique and get the 4 LDL_f monitors.

Compensation Constraints

- An order cannot be canceled anymore if it is closed.

$\varphi_{canc} =$ 

- If **this happens**, then the customer has to pay a supplement:

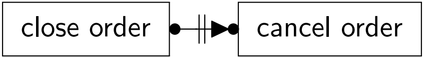
$\varphi_{dopay} =$ 

- Formally: $\{[\varphi_{canc}]_{RV} = \mathit{false}\} \rightarrow \varphi_{dopay}$
- In LDL_f: $(\langle \mathsf{pref}_{\neg\varphi_{canc}} \rangle \mathit{end} \wedge \neg \langle \mathsf{pref}_{\varphi_{canc}} \rangle \mathit{end}) \rightarrow \varphi_{dopay}$

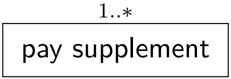
Compensation Constraints

unibz

- An order cannot be canceled anymore if it is closed.

$\varphi_{canc} =$ 

- If **this happens**, then the customer has to pay a supplement:

$\varphi_{dopay} =$ 

- Formally: $\{[\varphi_{canc}]_{RV} = \mathit{false}\} \rightarrow \varphi_{dopay}$
- In LDL_f: $(\langle \mathsf{pref}_{\neg\varphi_{canc}} \rangle end \wedge \neg \langle \mathsf{pref}_{\varphi_{canc}} \rangle end) \rightarrow \varphi_{dopay}$

Observation

When the violation occurs, the compensation is monitored from the beginning of the trace: OK to *"compensate in advance"*.

- Trace `close order` $\rightarrow$ `pay supplement` $\rightarrow$ `cancel order` is OK.

Metaconstraints Revisited

Business metaconstraint with temporal consequence

1. Take Φ_{pre} and Ψ_{exp} as before.
2. Compute ρ: regular expression denoting those paths that satisfy Φ_{pre}
3. Make ρ part of the compensation:

$$\Phi_{pre} \rightarrow \; [\rho] \; \Psi_{exp}$$

Metaconstraints Revisited

unibz

Business metaconstraint with temporal consequence

1. Take Φ_{pre} and Ψ_{exp} as before.
2. Compute ρ: regular expression denoting those paths that satisfy Φ_{pre}
3. Make ρ part of the compensation:

$$\Phi_{pre} \rightarrow \boxed{[\rho]}\ \Psi_{exp}$$

Ψ_{exp} has now to be enforced **after** Φ_{pre} becomes true.

Compensation Revisited

unibz

- An order cannot be canceled anymore if it is closed.

$$\varphi_{canc} = \boxed{\text{close order}} \bullet\!\!-\!\!\|\!\!\blacktriangleright\!\!\bullet \boxed{\text{cancel order}}$$

- **After** this happens, then the customer has to pay a supplement:

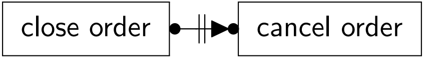
$$\varphi_{dopay} = \overset{1..*}{\boxed{\text{pay supplement}}}$$

- Formally:

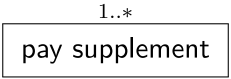
$$\{[\varphi_{canc}]_{RV} = \mathit{false}\} \rightarrow \left[re_{\{[\varphi_{canc}]_{RV} = \mathit{false}\}}\right] \varphi_{dopay}$$

Compensation Revisited

- An order cannot be canceled anymore if it is closed.

$\varphi_{canc} =$ 

- **After** this happens, then the customer has to pay a supplement:

$\varphi_{dopay} =$ 

- Formally:

$$\{[\varphi_{canc}]_{RV} = \mathit{false}\} \rightarrow \boxed{[re_{\{[\varphi_{canc}]_{RV} = \mathit{false}\}}]} \varphi_{dopay}$$

All traces violating φ_{canc}

From LDL_f to NFA

unibz

Direct calculation of NFA corresponding to LDL_f formula φ

Algorithm

algorithm $\text{LDL}_f\text{2NFA}()$
input LTL_f formula φ
output NFA $A_\varphi = (2^{\mathcal{P}}, \mathcal{S}, \{s_0\}, \varrho, \{s_f\})$
$s_0 \leftarrow \{\text{"}\varphi\text{"}\}$ ▷ single initial state
$s_f \leftarrow \emptyset$ ▷ single final state
$\mathcal{S} \leftarrow \{s_0, s_f\}$, $\varrho \leftarrow \emptyset$
while ($\mathcal{S}$ or ϱ change) **do**
  if ($q \in \mathcal{S}$ and $q' \models \bigwedge_{(\text{"}\psi\text{"} \in q)} \delta(\text{"}\psi\text{"}, \Theta)$)
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q'\}$ ▷ update set of states
    $\varrho \leftarrow \varrho \cup \{(q, \Theta, q')\}$ ▷ update transition relation

Note

- Standard NFA.
- No detour to Büchi automata.
- Easy to code.
- Implemented!

Auxiliary rules

$$\delta(\text{"}tt\text{"}, \Pi) = \mathit{true}$$

$$\delta(\text{"}f\!f\text{"}, \Pi) = \mathit{false}$$

$$\delta(\text{"}\phi\text{"}, \Pi) = \begin{cases} \mathit{true} \text{ if } \Pi \models \phi \\ \mathit{false} \text{ if } \Pi \not\models \phi \end{cases} \quad (\phi \text{ propositional})$$

$$\delta(\text{"}\varphi_1 \wedge \varphi_2\text{"}, \Pi) = \delta(\text{"}\varphi_1\text{"}, \Pi) \wedge \delta(\text{"}\varphi_2\text{"}, \Pi)$$

$$\delta(\text{"}\varphi_1 \vee \varphi_2\text{"}, \Pi) = \delta(\text{"}\varphi_1\text{"}, \Pi) \vee \delta(\text{"}\varphi_2\text{"}, \Pi)$$

$$\delta(\text{"}\langle\phi\rangle\varphi\text{"}, \Pi) = \begin{cases} \text{"}\varphi\text{"} \text{ if } \mathit{last} \notin \Pi \text{ and } \Pi \models \phi & (\phi \text{ propositional}) \\ \delta(\text{"}\varphi\text{"}, \epsilon) \text{ if } \mathit{last} \in \Pi \text{ and } \Pi \models \phi \\ \mathit{false} \text{ if } \Pi \not\models \phi \end{cases}$$

$$\delta(\text{"}\langle\psi?\rangle\varphi\text{"}, \Pi) = \delta(\text{"}\psi\text{"}, \Pi) \wedge \delta(\text{"}\varphi\text{"}, \Pi)$$

$$\delta(\text{"}\langle\rho_1 + \rho_2\rangle\varphi\text{"}, \Pi) = \delta(\text{"}\langle\rho_1\rangle\varphi\text{"}, \Pi) \vee \delta(\text{"}\langle\rho_2\rangle\varphi\text{"}, \Pi)$$

$$\delta(\text{"}\langle\rho_1; \rho_2\rangle\varphi\text{"}, \Pi) = \delta(\text{"}\langle\rho_1\rangle\langle\rho_2\rangle\varphi\text{"}, \Pi)$$

$$\delta(\text{"}\langle\rho^*\rangle\varphi\text{"}, \Pi) = \begin{cases} \delta(\text{"}\varphi\text{"}, \Pi) & \text{if } \rho \text{ is test-only} \\ \delta(\text{"}\varphi\text{"}, \Pi) \vee \delta(\text{"}\langle\rho\rangle\langle\rho^*\rangle\varphi\text{"}, \Pi) & \text{o/w} \end{cases}$$

$$\delta(\text{"}[\phi]\varphi\text{"}, \Pi) = \begin{cases} \text{"}\varphi\text{"} \text{ if } \mathit{last} \notin \Pi \text{ and } \Pi \models \phi & (\phi \text{ propositional}) \\ \delta(\text{"}\varphi\text{"}, \epsilon) \text{ if } \mathit{last} \in \Pi \text{ and } \Pi \models \phi & (\phi \text{ propositional}) \\ \mathit{true} \text{ if } \Pi \not\models \phi \end{cases}$$

$$\delta(\text{"}[\psi?]\varphi\text{"}, \Pi) = \delta(\text{"}\mathit{nnf}(\neg\psi)\text{"}, \Pi) \vee \delta(\text{"}\varphi\text{"}, \Pi)$$

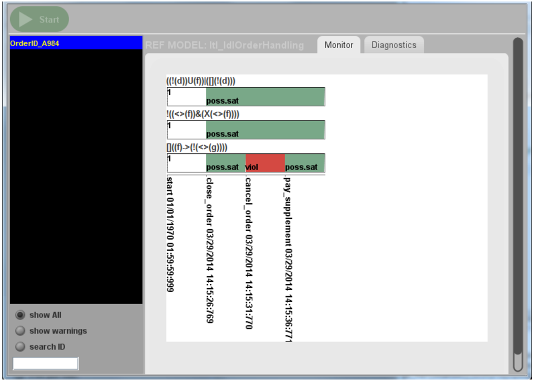
$$\delta(\text{"}[\rho_1 + \rho_2]\varphi\text{"}, \Pi) = \delta(\text{"}[\rho_1]\varphi\text{"}, \Pi) \wedge \delta(\text{"}[\rho_2]\varphi\text{"}, \Pi)$$

$$\delta(\text{"}[\rho_1; \rho_2]\varphi\text{"}, \Pi) = \delta(\text{"}[\rho_1][\rho_2]\varphi\text{"}, \Pi)$$

$$\delta(\text{"}[\rho^*]\varphi\text{"}, \Pi) = \begin{cases} \delta(\text{"}\varphi\text{"}, \Pi) & \text{if } \rho \text{ is test-only} \\ \delta(\text{"}\varphi\text{"}, \Pi) \wedge \delta(\text{"}[\rho][\rho^*]\varphi\text{"}, \Pi) & \text{o/w} \end{cases}$$

Implemented in ProM!

unibz



Approach

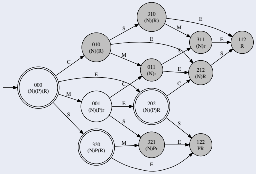
1. Input $\text{LTL}_f/\text{LDL}_f$ constraints and metaconstraints.
2. Produce the corresponding RV LDL_f monitoring formulae.
3. Apply the direct algorithm and get the corresponding NFAs.
4. (Incrementally) run NFAs the monitored trace.

Connection with Colored Automata

Colored Automata [BPM2011]

Ad-hoc technique for monitoring LTL_f formulae according to RV-LTL.

1. Color states of each local automaton according to the 4 RV-LTL truth values.
2. Combine colored automata into a global colored automaton.



Why is step 1 correct?

1. Take the LTL_f formula φ of a constraint.
2. Produce the 4 corresponding LDL_f monitoring formulae.
3. Generate the 4 corresponding NFAs.
4. Determinize them $\rightsquigarrow$ they are identical but with $\neq$ acceptance states!
5. Hence they can be combined into a unique colored local DFA.

Conclusion

- Focus on finite traces.
- Avoid unneeded detour to infinite traces.
- LDL_f: essentially, the maximal expressive logic for finite traces with good computational properties ($\equiv$ MSO).
- Monitoring is a key problem.
- LDL_f goes far beyond DECLARE.
- LDL_f captures monitors directly as formulae.
  - Clean.
  - Meta-constraints.
- Implemented in ProM!

Future work: declarative, data-aware processes.