DIPARTIMENTO DI
INFORMATICA, SISTEMISTICA E
COMUNICAZIONE

Mixing Time and Uncertainty

A Tale of Superpositions

Rafael Peñaloza

The World, February 14th, 2024

Università di Milano-Bicocca

Business Processes

There is a need to formally represent (business) processes

- understand the golden standard
- avoid unwanted behaviour
- guarantee a service

Expressed in the form of constraints

A Specification Language

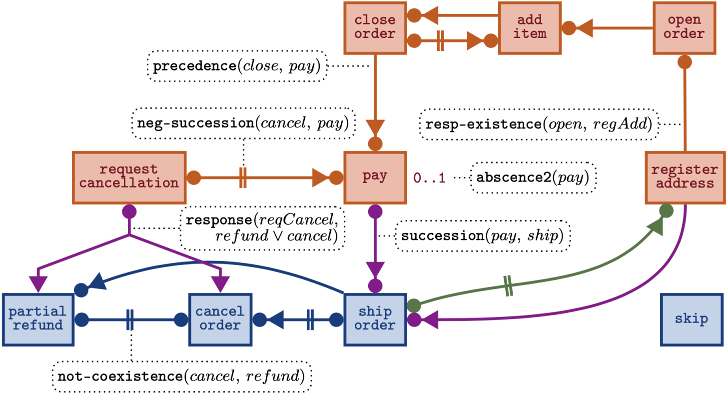
DECLARE is a simple specification language

Restricts potential (temporal) relationships between actions

Graphical, easy to read, representation

Only simple constraints available

A Declare Specification



Example: Order Handling Process

- only paid orders should be shipped
- every paid order should be delivered
- orders must be returned before being reimbursed
- orders cannot be open and closed
- empty orders cannot be paid
- ...

The Backbone

DECLARE constraints are merely $\mathsf{LTL_f}$ formulas

$$\varphi := x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \varphi \mathcal{U} \varphi$$

Syntactically as LTL, but over finite time

(processes always terminate)

In $\mathsf{LTL_f}$, $\neg\bigcirc\varphi \not\equiv \bigcirc\neg\varphi$

Usual abbreviations: $\Diamond\varphi$, $\Box\varphi$

DECLARE Reasoning

A DECLARE specification is a big LTL_f formula

Use standard reasoning techniques to check

- satisfiability (is the specification possible?)
- entailment (is a property guaranteed?)
- safety (is an unwanted situation avoided?)

From Theory to Practice

We have these nice specification language but:

- how do we construct a specification?
- does the process comply with its specification?

We have to look at logs

The Real World Doesn't Like Us

It is common to deviate from a deterministic specification

- every shipped order must have been paid
  (deviation by design)
- every paid order must be delivered
  (deviation by uncertainty)
- only delivered orders can be returned
  (deviation by execution)

but we should quantify these deviations

and have guarantees under uncertainty

Probabilistic Temporal Logic

The goal is to extend LTL_f (DECLARE) to handle probabilities

Language should be clear and simple

Reasoning should be (probabilistically) sound

Enter $\mathsf{PLTL_f}$

$\mathsf{PLTL_f}$ extends $\mathsf{LTL_f}$ with a new

temporal probabilistic constructor

$$\varphi := x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \circledcirc_{\bowtie p}\varphi$$

$$p \in [0,1], \qquad \bowtie \text{ an order relation } (<, =, \geq, \ldots)$$

Probabilities apply to the next point in time

(for technical reasons)

Examples

$\circledcirc_{\geq 0.9} x$: the probability of seeing x next is at least 0.9

$\circledcirc_{\geq 0.95} \Diamond x$:
the probability of eventually seeing x is at least 0.95

$\Box \circledcirc_{\leq 0.01} x$:
at every point in time, the probability of x is at most 0.01

$\circledcirc_{\geq 0.5} x \wedge \circledcirc_{\leq 0.9} \neg y$

Superposition Semantics

A trace is simultaneously in all its configurations
until a distinguishing observation is made

As time evolves, the trace may fall into different
superimposed states

The notion of an interpretation should take
all these superpositions into account

LTL_f Semantics

In LTL_f, an interpretation is a finite sequence of propositional valuations

Satisfaction of formulas defined at each point depending on the future

$\Diamond x$

From $\mathrm{LTL_f}$ to $\mathrm{PLTL_f}$ Semantics

$\circledcirc_{\bowtie p}\varphi$ tells us something about the probability of φ

but also about $\neg\varphi$

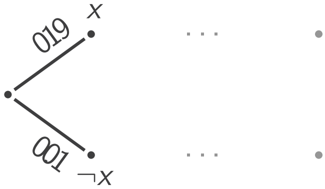
$\circledcirc_{\geq 0.9}x$ means also $\circledcirc_{<0.1}\neg x$

both possibilities must be considered

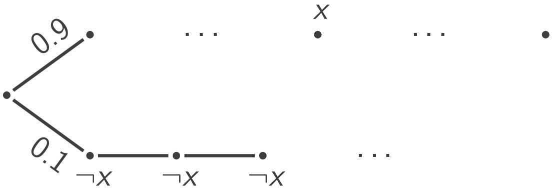
Interpretations branch to explore different situations

Example

$\circledcirc_{\geq 0.9} x$



$\circledcirc_{\geq 0.9} \Diamond x$

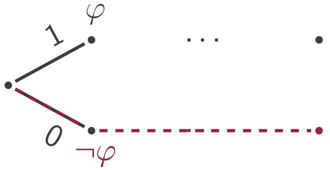


Handling Tautologies

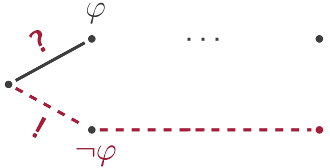
What happens if φ is a tautology?

it is true in every interpretation

$\circledcirc_{\geq 0.9}\varphi$



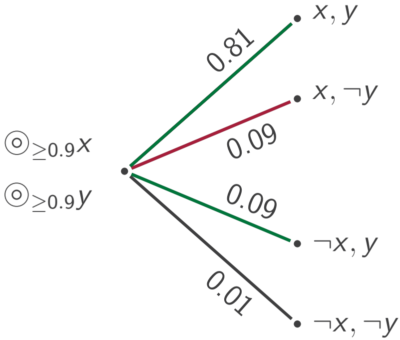
$\circledcirc_{\leq 0.9}\varphi$



Example II: Multiple Superpositions

Consider $\circledcirc_{\geq 0.9} x \wedge \circledcirc_{\geq 0.9} y$

there are four possibilities for the next point in time

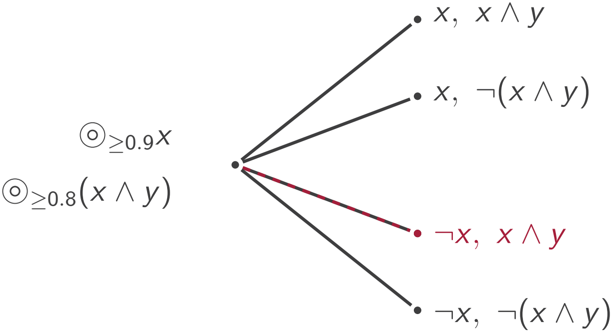


Are these probabilities always feasible?

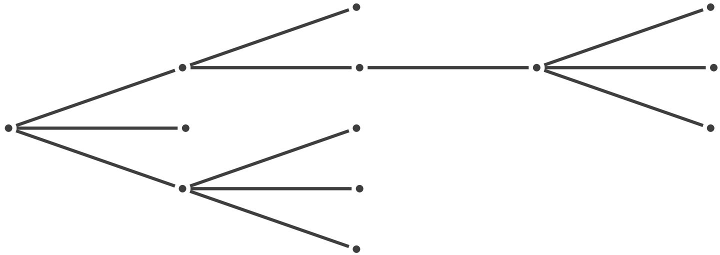
No Independence Assumption

Probabilistic constraints may have complex dependencies

e.g. $\circledcirc_{\geq 0.9} x \wedge \circledcirc_{\geq 0.8}(x \wedge y)$



Tree-Shaped Interpretations



Nodes labelled with propositional valuations,

edges labelled in $[0, 1]$

Each branch is a representative for a trace type

Multiple Superpositions

We cannot require all branches to exist

Non-deterministically guess which branches to have

Try to solve numerical and logical constraints

- a system of inequalities
- recursion

Any solution is a model

Hand Waving

The actual approach is quite technical

but the intuition is simple

Satisfiability

Satisfiability can be decided through a tree automaton

Transitions are constructed by solving systems of inequalities (pre-processing)

Decision through a standard emptiness test
does the automaton accept at least one tree?

High Complexity

If there are n probabilistic constraints
the maximal branching in a model is 2^n

I.e., the automaton is huge
but emptiness is polynomial on automata size

PLTL_f satisfiability is in ExpSpace (*)

Probabilistic Reasoning

Satisfiability does not take probabilities into account

except for solution existence

But we may want to know probabilities of entailments

- how likely it is to see a given behaviour?
- is the business process safe?
- what is the most likely trace?

What is that probability?

consider optimistic and pessimistic views

Most Likely Trace (Optimistically)

A (tree) model is a compact description of
(linear) temporal interpretations

We can see each branch as a trace (execution)

The most likely trace is the branch maximising probability
over all models

Expand to most likely language

Flattening

The tree automaton is transformed into a weighted string automaton

Its behaviour corresponds to the likelihood of the mlt

A further weight-removal yields an NFA accepting the MLL

All computable in PSPACE (after a pre-processing)

Optimistic (and Pessimistic) Reasoning

How likely it is to observe a property?

Must accumulate over all adequate branches!

Restricting to only the relevant paths

Similar flattening (although more technical)

A Special Case: $\mathbf{PLTL_f^0}$

Process models are defined through constraints
specified at the beginning of the process

They refer about specific traces
uncertainty about its shape with linear execution

Root is the only branching point multiple-world semantics

Trace Mining

Where do the probabilities come from?

We simply read them from process logs

Impose safe constraints, which conform with reality

. . . or do they?

Looking Ahead

Logs are not complete they are samples of executions

We consider most likely estimators not fully informative

Need to develop a statistical temporal logic:

- probability distributions
- quantiles
- predictions
- ...

rafael.penaloza@unimib.it