Reactive Synthesis for DECLARE Process Models

Luca Geatti *
Marco Montali
Andrey Rivkin

On the Effectiveness of Temporal Logics on Finite Traces in AI
AAAI 2023 Spring Symposium Series, March 27-29, 2023, Hyatt Regency San Francisco Airport, CA, USA

Background

DECLARE

- Pattern-based declarative process modelling language.
- It is used for the modelling of business processes.
- It relies on a pre-defined set of unary and binary templates.
- Each pattern is formalized in LTLf.

Pattern	LTLf **formalization**
existence(p)	$\mathsf{F}(p)$
absence2(p)	$\neg\mathsf{F}(p \wedge \mathsf{XF}(p))$
choice(p, q)	$\mathsf{F}(p) \vee \mathsf{F}(q)$
exc-choice(p, q)	$(\mathsf{F}(p) \vee \mathsf{F}(q)) \wedge \neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$
resp-existence(p, q)	$\mathsf{F}(p) \rightarrow \mathsf{F}(q)$
coexistence(p, q)	$\mathsf{F}(p) \leftrightarrow \mathsf{F}(q)$
response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{F}(q))$
precedence(p, q)	$(\neg q)\,\mathsf{W}\,(p)$
succession(p, q)	$\mathsf{G}(p \rightarrow \mathsf{F}(q)) \wedge (\neg q)\,\mathsf{W}\,(p)$
alt-response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\,\mathsf{U}\,q))$
alt-precedence(p, q)	$((\neg q)\,\mathsf{W}\,p) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\,\mathsf{W}\,p))$
alt-succession(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\,\mathsf{U}\,q)) \wedge ((\neg q)\,\mathsf{W}\,p) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\,\mathsf{W}\,p))$
chain-response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}(q))$
chain-precedence(p, q)	$\mathsf{G}(\mathsf{X}(q) \rightarrow p)$
chain-succession(p, q)	$\mathsf{G}(p \leftrightarrow \mathsf{X}(q))$
not-coexistence(p, q)	$\neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$
neg-succession(p, q)	$\mathsf{G}(p \rightarrow \neg\mathsf{F}(q))$
neg-chain-succession(p, q)	$\mathsf{G}(p \rightarrow \widetilde{\mathsf{X}}(\neg q)) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}(\neg p))$

Background

DECLARE

- It is interpreted over simple traces
- Simple trace

$$\sigma = \langle \sigma_0, \ldots, \sigma_n \rangle \in (2^{\Sigma})^+$$

$$\text{such that } |\sigma_i| = 1 \text{ for any } 0 \leq i < |\sigma|$$

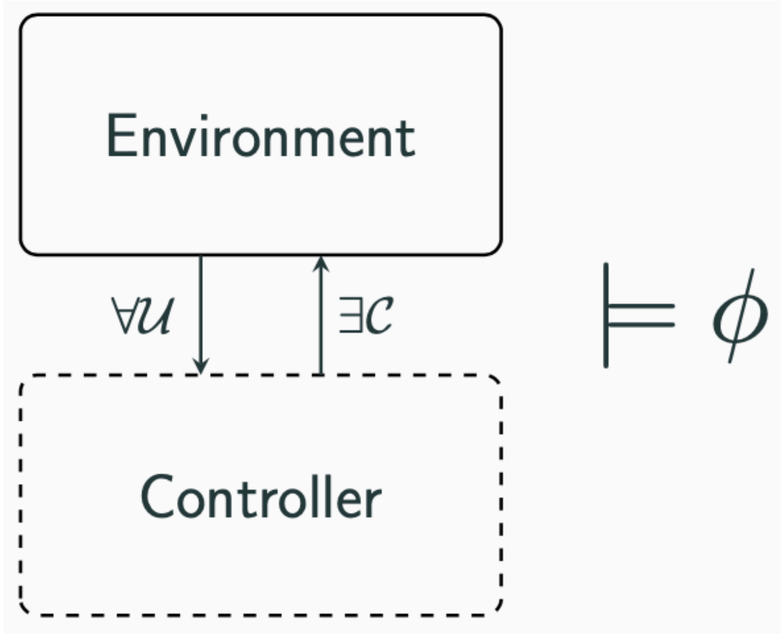
- DECLARE patterns are conjunctively related. DECLARE specification

Phi := P1 & P2 & ... & Pn

Pattern	LTLf formalization
existence(p)	$\mathsf{F}(p)$
absence2(p)	$\neg\mathsf{F}(p \wedge \mathsf{XF}(p))$
choice(p, q)	$\mathsf{F}(p) \vee \mathsf{F}(q)$
exc-choice(p, q)	$(\mathsf{F}(p) \vee \mathsf{F}(q)) \wedge \neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$
resp-existence(p, q)	$\mathsf{F}(p) \rightarrow \mathsf{F}(q)$
coexistence(p, q)	$\mathsf{F}(p) \leftrightarrow \mathsf{F}(q)$
response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{F}(q))$
precedence(p, q)	$(\neg q)\ \mathsf{W}\ (p)$
succession(p, q)	$\mathsf{G}(p \rightarrow \mathsf{F}(q)) \wedge (\neg q)\ \mathsf{W}\ (p)$
alt-response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\ \mathsf{U}\ q))$
alt-precedence(p, q)	$((\neg q)\ \mathsf{W}\ p) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\ \mathsf{W}\ p))$
alt-succession(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\ \mathsf{U}\ q)) \wedge ((\neg q)\ \mathsf{W}\ p) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\ \mathsf{W}\ p))$
chain-response(p, q)	$\mathsf{G}(p \rightarrow \mathsf{X}(q))$
chain-precedence(p, q)	$\mathsf{G}(\mathsf{X}(q) \rightarrow p)$
chain-succession(p, q)	$\mathsf{G}(p \leftrightarrow \mathsf{X}(q))$
not-coexistence(p, q)	$\neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$
neg-succession(p, q)	$\mathsf{G}(p \rightarrow \neg\mathsf{F}(q))$
neg-chain-succession(p, q)	$\mathsf{G}(p \rightarrow \widetilde{\mathsf{X}}(\neg q)) \wedge \mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}(\neg p))$

Background

Reactive Synthesis and Realizability



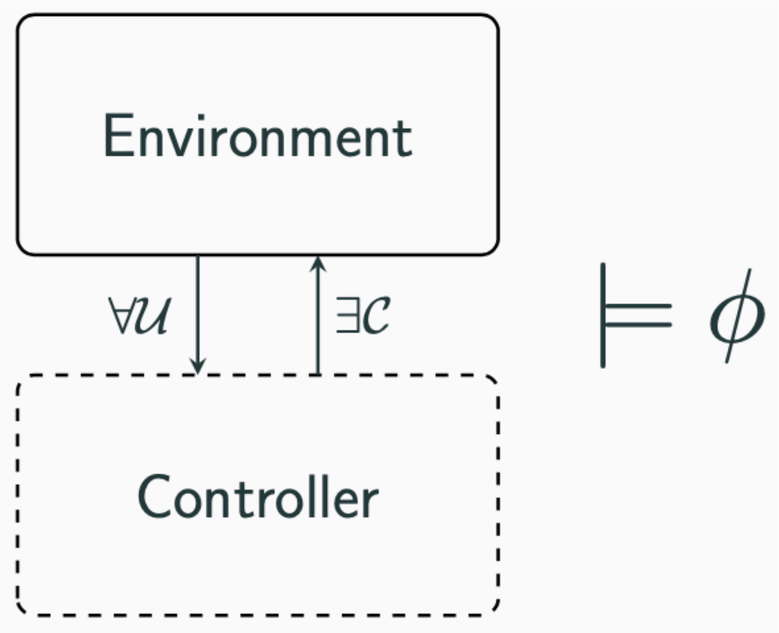
Definition 3 (Strategy). *A* strategy *for* Controller *is a function* $s : (2^{\mathcal{U}})^+ \to 2^{\mathcal{C}}$ *that, for any finite sequence* $\mathsf{U} = \langle \mathsf{U}_0, \ldots, \mathsf{U}_n \rangle$ *of choices by* Environment, *determines the choice* $\mathsf{C}_n = s(\mathsf{U})$ *of* Controller.

Given an infinite sequence of choices by Environment $\mathsf{U} = \langle \mathsf{U}_0, \mathsf{U}_1, \ldots \rangle \in (2^{\mathcal{U}})^\omega$, we denote by $\mathtt{res}(s, \mathsf{U}) = \langle \mathsf{U}_0 \cup s(\langle \mathsf{U}_0 \rangle), \mathsf{U}_1 \cup s(\langle \mathsf{U}_0, \mathsf{U}_1 \rangle), \ldots \rangle$ the trace resulting from reacting to U according to s. Realizability is usually modeled as a two-player game between Environment and Controller, who try to respectively violate and fulfill the given formula.

Definition 4 (Realizability and Reactive Synthesis for LTLf). *Let* ϕ *be a* LTLf *formula over* Σ. ϕ *is* realizable *(over finite words) iff there exists a strategy* $s : (2^{\mathcal{U}})^+ \to 2^{\mathcal{C}}$ *s.t., for any infinite sequence* $\mathsf{U} = \langle \mathsf{U}_0, \mathsf{U}_1, \ldots \rangle$ *in* $(2^{\mathcal{U}})^\omega$ *there exists* $k \in \mathbb{N}$ *for which* $\mathtt{res}(s, \mathsf{U})_{[0,k]} \models \phi$. *If* ϕ *is realizable,* reactive synthesis *is the problem of computing such a strategy* s.[2]

Background

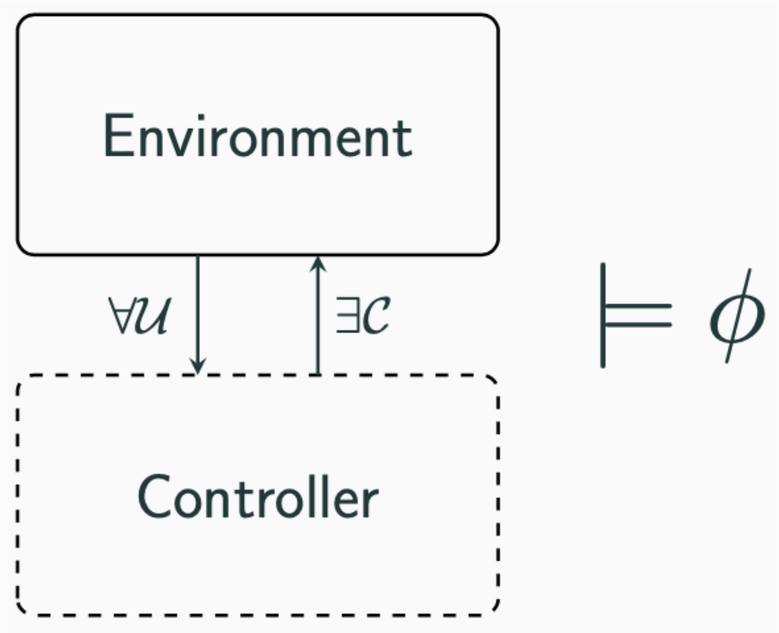
Reactive Synthesis



- History:
  - originally proposed by Church
  - solved for S1S by Landweber and Buchi (nonelementary complexity)
  - solved for LTL by Pnueli and Rosner (doubly exponential complexity)
    - requires Safra's determinization algorithm
  - Safraless approach & symbolic techniques (Kupferman & Vardi)

Background

Reactive Synthesis



- Complexity:
  - LTL : 2EXPTIME-complete
  - LTLf : 2EXPTIME-complete
  - SafetyLTL : 2EXPTIME-complete
  - ppLTL : EXPTIME-complete

Contributions

1. We define the Reactive Synthesis problem for DECLARE
   - assume-guarantee paradigm
2. Naive algorithm (reduction to LTLf reactive synthesis)
   - doubly exponential time complexity
3. Efficient algorithm
   - singly exponential time complexity
   - based on the "pastification" of all DECLARE patterns
4. Symbolic algorithm
   - new encoding of pure past LTL formulas into symbolic DFAs

Reactive Synthesis for DECLARE

Definition

- Reactive Synthesis of LTLf over simple traces

General Traces

$$s : (2^{\mathcal{U}})^+ \rightarrow 2^{\mathcal{C}}$$

Simple Traces

$$s : (\mathcal{U})^+ \rightarrow \mathcal{C}$$

strict alternation

Reactive Synthesis for DECLARE

Definition

- *There is background knowledge on how Environment operates. In particular, in BPM the external stakeholders participate to the process in a constrained way: when it is their turn, they take (arbitrary) decisions on which next action to trigger but only on the subset of all actions made available by the information system supporting the enactment of the process [Dumas et al., 2018]*

- Assume-Guarantee paradigm:

Phi_E := {P1, ... , Pm}

Phi_C := {P1, ... , Pn}

- Definition of DECLARE realizability

We say that the pair (Phi_E, Phi_C) is realizable iff Phi_E -> Phi_C is realizable over simple traces.

Algorithms

A Naive Algorithm

for DECLARE reactive synthesis

- Auxiliary formulas for forcing simple trace + strict alternation

$$\texttt{simple}_{\texttt{Env}}(\mathcal{U}) \coloneqq \bigvee_{u \in \mathcal{U}} u \wedge \mathsf{G}(\bigvee_{u \in \mathcal{U}} u \rightarrow (\bigwedge_{u \neq u' \in \mathcal{U}} \neg(u \wedge u') \wedge \widetilde{\mathsf{X}}(\bigwedge_{u \in \mathcal{U}} \neg u \wedge \widetilde{\mathsf{X}} \bigvee_{u \in \mathcal{U}} u)))$$

$$\texttt{simple}_{\texttt{Con}}(\mathcal{C}) \coloneqq \bigwedge_{c \in \mathcal{C}} \neg c \wedge \mathsf{G}(\bigwedge_{c \in \mathcal{C}} \neg c \rightarrow \widetilde{\mathsf{X}}(\bigvee_{c \in \mathcal{C}} c \wedge \bigwedge_{c \neq c' \in \mathcal{C}} \neg(c \wedge c') \wedge \widetilde{\mathsf{X}} \bigwedge_{c \in \mathcal{C}} \neg c)))$$

- Reduction to LTLf reactive synthesis

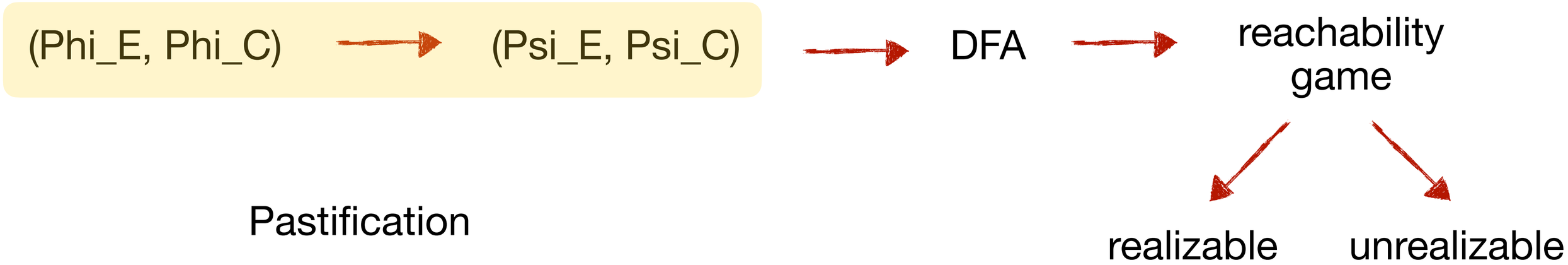
(ϕ_E, ϕ_C) is realizable

<=>

$\texttt{simple}_{\texttt{Con}}(\mathcal{C}) \wedge ((\texttt{simple}_{\texttt{Env}}(\mathcal{U}) \wedge \phi_E) \rightarrow \phi_C)$ is realizable

An Efficient Algorithm

for DECLARE reactive synthesis



Pastification of DECLARE patterns

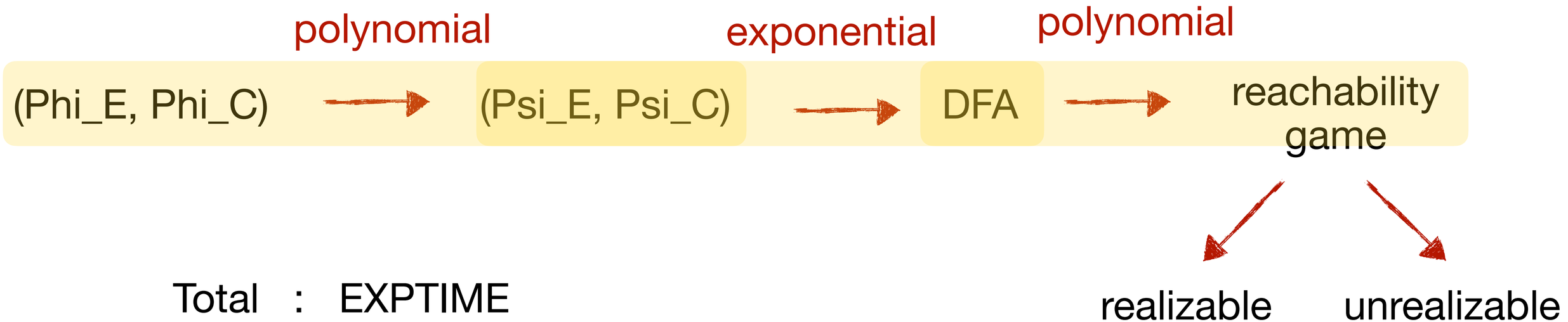
in polynomial space

- DECLARE is one of the very few logics that admits a polynomial-size pastification.
- We conjecture that a polynomial-size pastification does not exists even for LTL[X,F].

Pattern	LTLf **formalization**	**Pastification** ($\mathrm{past}(\cdot)$)
$\texttt{existence}(p)$	$\mathsf{F}(p)$	$\mathsf{O}(p)$
$\texttt{absence2}(p)$	$\neg\mathsf{F}(p \wedge \mathsf{XF}(p))$	$\mathsf{H}(p \rightarrow \mathsf{ZH}(\neg p))$
$\texttt{choice}(p, q)$	$\mathsf{F}(p) \vee \mathsf{F}(q)$	$\mathsf{O}(p \vee q)$
$\texttt{exc-choice}(p, q)$	$(\mathsf{F}(p) \vee \mathsf{F}(q)) \wedge$ $\neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$	$\mathsf{O}(p \vee q) \wedge$ $(\mathsf{H}(\neg p) \vee \mathsf{H}(\neg q))$
$\texttt{resp-existence}(p, q)$	$\mathsf{F}(p) \rightarrow \mathsf{F}(q)$	$\mathsf{H}(\neg p) \vee \mathsf{O}(q)$
$\texttt{coexistence}(p, q)$	$\mathsf{F}(p) \leftrightarrow \mathsf{F}(q)$	$(\mathsf{H}(\neg p) \vee \mathsf{O}(q)) \wedge$ $(\mathsf{H}(\neg q) \vee \mathsf{O}(p))$
$\texttt{response}(p, q)$	$\mathsf{G}(p \rightarrow \mathsf{F}(q))$	$q\,\mathsf{T}\,(\neg p \vee q)$
$\texttt{precedence}(p, q)$	$(\neg q)\,\mathsf{W}\,(p)$	$\mathsf{H}(q \rightarrow \mathsf{O}(p))$
$\texttt{succession}(p, q)$	$\mathsf{G}(p \rightarrow \mathsf{F}(q)) \wedge$ $(\neg q)\,\mathsf{W}\,(p)$	$p\,\mathsf{T}\,(\neg p \vee q) \wedge$ $\mathsf{H}(q \rightarrow \mathsf{O}(p))$
$\texttt{alt-response}(p, q)$	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\,\mathsf{U}\,q))$	$(p \vee q)\,\mathsf{T}\,(\neg p) \wedge \mathsf{H}(q \rightarrow \mathsf{Z}(q\,\mathsf{T}\,((p \wedge$ $\neg q) \rightarrow \mathsf{Z}(q\,\mathsf{T}\,\neg p))))$
$\texttt{alt-precedence}(p, q)$	$((\neg q)\,\mathsf{W}\,p) \wedge$ $\mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\,\mathsf{W}\,p))$	$\mathsf{H}(q \rightarrow \mathsf{O}(p)) \wedge$ $\mathsf{H}((q \wedge \neg p) \rightarrow$ $\mathsf{Z}(p\,\mathsf{T}\,(q \rightarrow (p\,\mathsf{T}\,(\neg p)))))$
$\texttt{alt-succession}(p, q)$	$\mathsf{G}(p \rightarrow \mathsf{X}((\neg p)\,\mathsf{U}\,q)) \wedge$ $((\neg q)\,\mathsf{W}\,p) \wedge$ $\mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}((\neg q)\,\mathsf{W}\,p))$	$\mathrm{past}(\texttt{alt-response}(p, q)) \wedge$ $\mathrm{past}(\texttt{alt-precedence}(p, q))$
$\texttt{chain-response}(p, q)$	$\mathsf{G}(p \rightarrow \mathsf{X}(q))$	$\neg p \wedge \mathsf{H}(\mathsf{Y}(p) \rightarrow q)$
$\texttt{chain-precedence}(p, q)$	$\mathsf{G}(\mathsf{X}(q) \rightarrow p)$	$\mathsf{H}(q \rightarrow \mathsf{Z}p)$
$\texttt{chain-succession}(p, q)$	$\mathsf{G}(p \leftrightarrow \mathsf{X}(q))$	$\mathrm{past}(\texttt{chain-response}(p, q)) \wedge$ $\mathrm{past}(\texttt{chain-precedence}(p, q))$
$\texttt{not-coexistence}(p, q)$	$\neg(\mathsf{F}(p) \wedge \mathsf{F}(q))$	$\mathsf{H}(\neg p) \vee \mathsf{H}(\neg q)$
$\texttt{neg-succession}(p, q)$	$\mathsf{G}(p \rightarrow \neg\mathsf{F}(q))$	$\mathsf{H}(\neg p) \vee (\neg q)\,\mathsf{S}\,(p \wedge \neg q \wedge \mathsf{ZH}(\neg p))$
$\texttt{neg-chain-succession}(p, q)$	$\mathsf{G}(p \rightarrow \widetilde{\mathsf{X}}(\neg q)) \wedge$ $\mathsf{G}(q \rightarrow \widetilde{\mathsf{X}}(\neg p))$	$\mathsf{H}(\mathsf{Y}(p) \rightarrow \neg q) \wedge$ $\mathsf{H}(\mathsf{Y}(q) \rightarrow \neg p)$

An Efficient Algorithm

Total Complexity



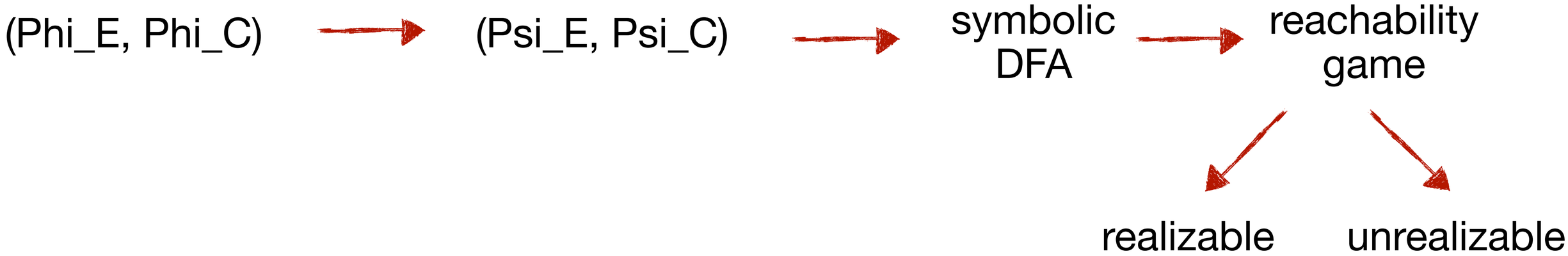
Total : EXPTIME

One exponential more efficient than LTLf

Symbolic Algorithm

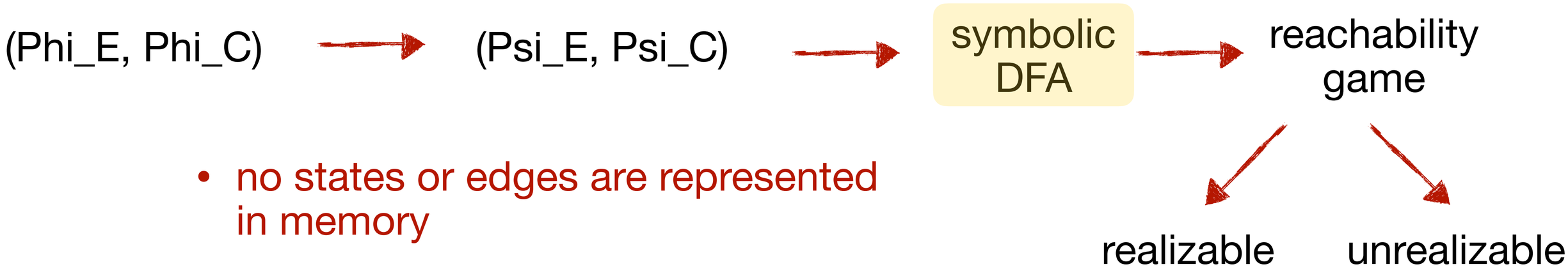
An Efficient Algorithm

for DECLARE reactive synthesis



An Efficient Algorithm

for DECLARE reactive synthesis

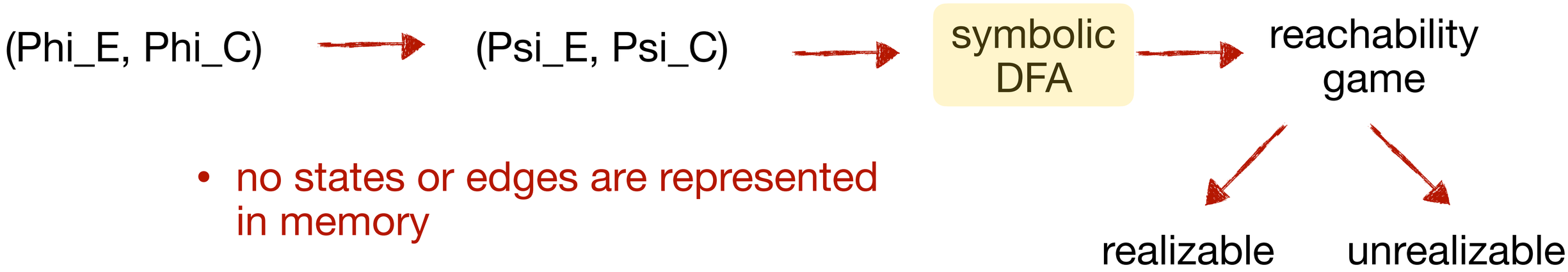


- no states or edges are represented in memory
- initial states, edges, and final states are represented using Boolean formulas

An Efficient Algorithm

for DECLARE reactive synthesis

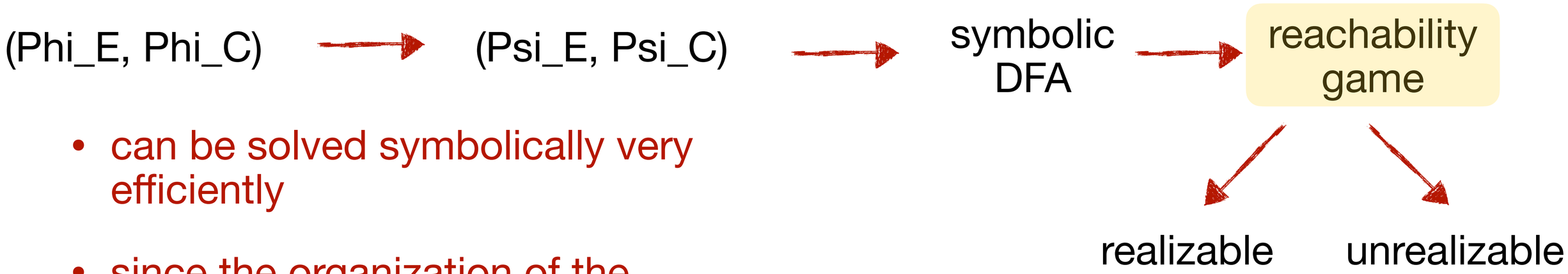
- NEW TECHNIQUE
- very simple construction
- uses classic expansion rules



- no states or edges are represented in memory
- initial states, edges, and final states are represented using Boolean formulas

An Efficient Algorithm

for DECLARE reactive synthesis

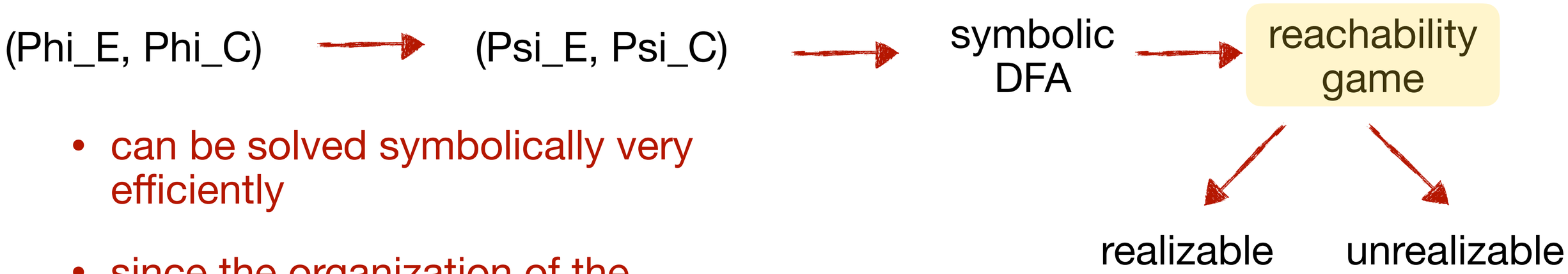


- can be solved symbolically very efficiently
- since the organization of the SYNTCOMP, a lot of optimized tools have been proposed to solve this problem

An Efficient Algorithm

for DECLARE reactive synthesis

- we reduced DECLARE reactive synthesis to a problem for which there are very efficient tools



- can be solved symbolically very efficiently
- since the organization of the SYNTCOMP, a lot of optimized tools have been proposed to solve this problem

Future Work

on DECLARE reactive synthesis

Implementation in a tool

+

Worst-case complexity (EXPTIME-completeness)

+

...