A Journey from LTLf Satisfiability to Synthesis

Jianwen Li
jwli@sei.ecnu.edu.cn
East China Normal University, Shanghai, China

March 28, 2023

Collaborated with Ofer Strichman and Moshe Vardi

Linear Temporal Logic

- First introduced to Computer Science by A. Pnueli in 1977
- Formal verification (over infinite traces: LTL)
- AI (over finite traces : LTLf) [IJCAI 13]

Linear Temporal Logic

Syntax for LTL and LTLf:

$$\varphi \coloneqq p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi \cup \varphi \mid \varphi\ R\ \varphi \mid G\varphi \mid F\varphi$$

- $\neg\,(\varphi_1 \mathrm{U} \varphi_2) \equiv \neg\varphi_1 R \neg\varphi_2$
- $\neg\,(X\varphi) \equiv \neg\, \mathrm{N}\, \neg\varphi$ (weak Next), for LTLf only
- $\mathrm{F}\varphi \equiv true\ U\ \varphi$
- $\mathrm{G}\varphi \equiv \mathrm{false}\ \mathrm{R}\ \varphi$

Linear Temporal Logic

Semantics for LTL (LTLf)

- Let δ be a trace with $|\delta| = n$ $(n > 0)$
  - $\delta \vDash \mathrm{p}$ if $\mathrm{p} \in \delta[0]$
  - $\delta \vDash \neg\varphi$ if $\delta \nvDash \varphi$
  - $\delta \vDash \varphi_1 \wedge \varphi_2$ if $\delta \vDash \varphi_1$ and $\delta \vDash \varphi_2$
  - $\delta \vDash \mathrm{X}\varphi$ if $n > 1$ and $\delta_1 \nvDash \varphi$
  - $\delta \vDash \varphi_1 \mathrm{U} \varphi_2$ *if* $\exists i \geq 0. \sigma_i \vDash \varphi_2$ *holds, and* $\forall 0 \leq j < i. \sigma_j \vDash \varphi_1$ *holds.*
- LTL semantics: $n = \infty$
- LTLf semantics: $n < \infty$

LTL vs. LTLf

- X true is always true in LTL, but not in LTLf
- (a $\wedge$ X true) $\not\equiv$ a in LTLf
- $\neg$ Xφ $\not\equiv$ X $\neg\varphi$ in LTLf ($\neg$ Xφ $\equiv$ N $\neg\varphi$)
- GXφ is unsatisfiable in LTLf

LTLf Satisfiability

- Given an LTLf formula φ, is there a non-empty finite trace δ such that $\delta \models \varphi$?

- G a is satisfiable
- G X a is unsatisfiable
- GF a ∧ GF ¬a is unsatisfiable

LTLf Synthesis

- Given an LTLf formula φ with the <$\mathcal{X}$,$\mathcal{Y}$> variable partition, is there a winning strategy $f: (2^{\mathcal{X}})^* \rightarrow 2^{Y}$ such that f will eventually produce a satisfiable trace of φ by interacting between the input ($\mathcal{X}$) and output ($\mathcal{Y}$) variables.

- We consider system-first synthesis
- G (a -> b) is realizable where $\mathcal{X}$={a} and $\mathcal{Y}$={b}
- G (a ∧ b) is unrealizable where $\mathcal{X}$={a} and $\mathcal{Y}$={b}

Satisfiability and Realizability (Synthesis)

- Both are fundamental problems for LTLf
- LTLf synthesis becomes popular due to its application to planning
- Satisfiability is easier than realizability in both theory and practice
- Question: Can we solve LTLf realizability via satisfiability?

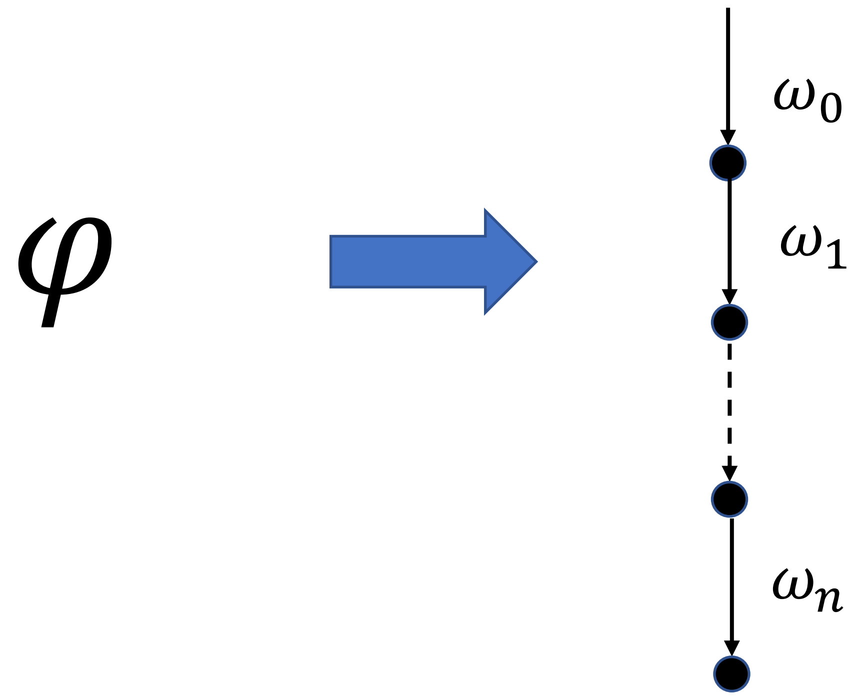
Satisfiability and Realizability (Synthesis)

- Both are fundamental problems for LTLf
- LTLf synthesis becomes popular due to its application to planning
- Satisfiability is easier than realizability in both theory and practice
- Question: Can we solve LTLf realizability via satisfiability?

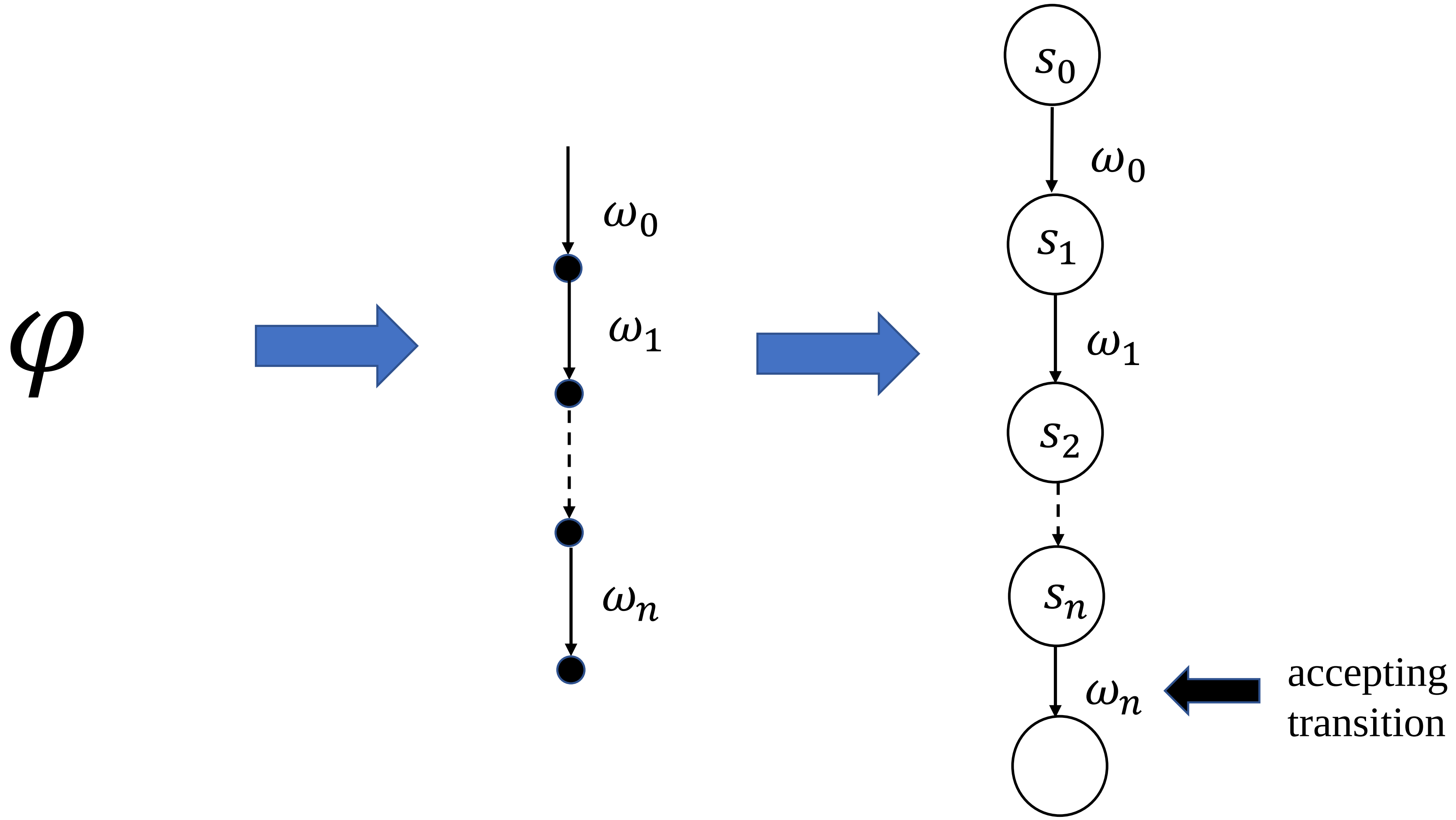
Yes!

Syn-SAT: A high-level description of the algorithm

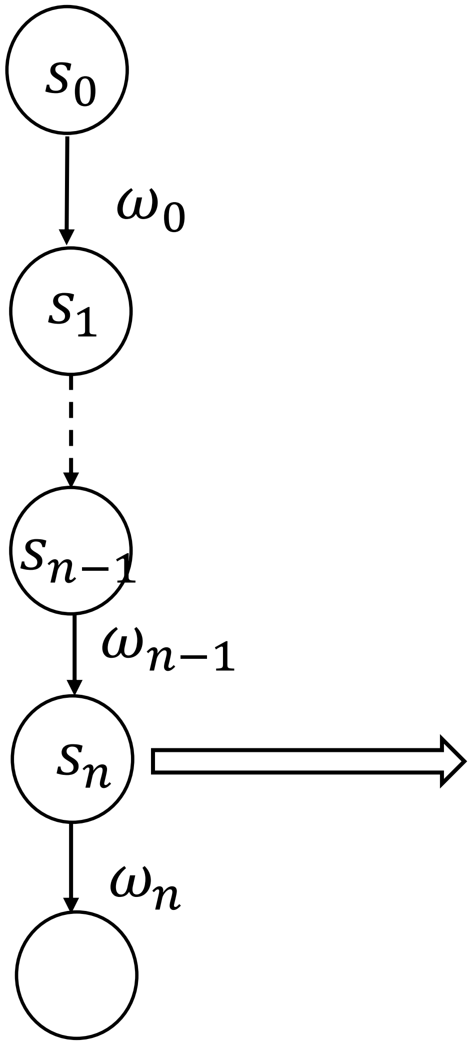
Step 1: Find a satisfiable trace



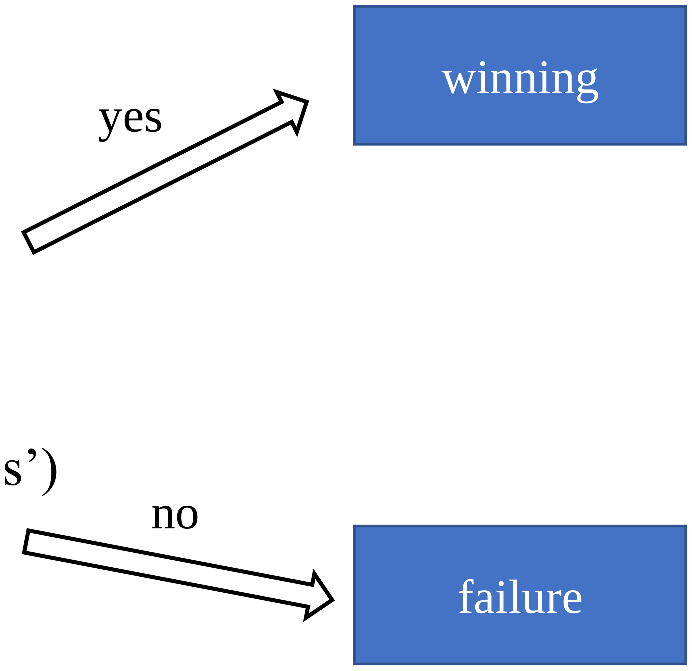
Step 2: Find a satisfiable run via progression



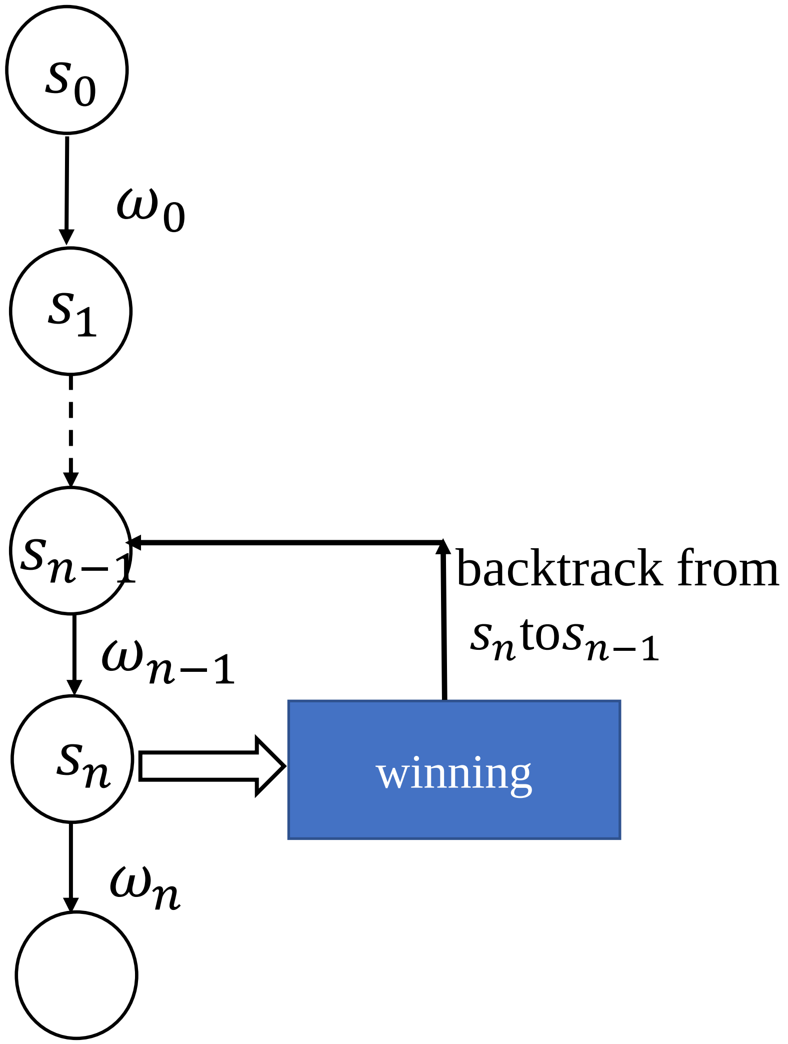
Step 3: check winning/failure states

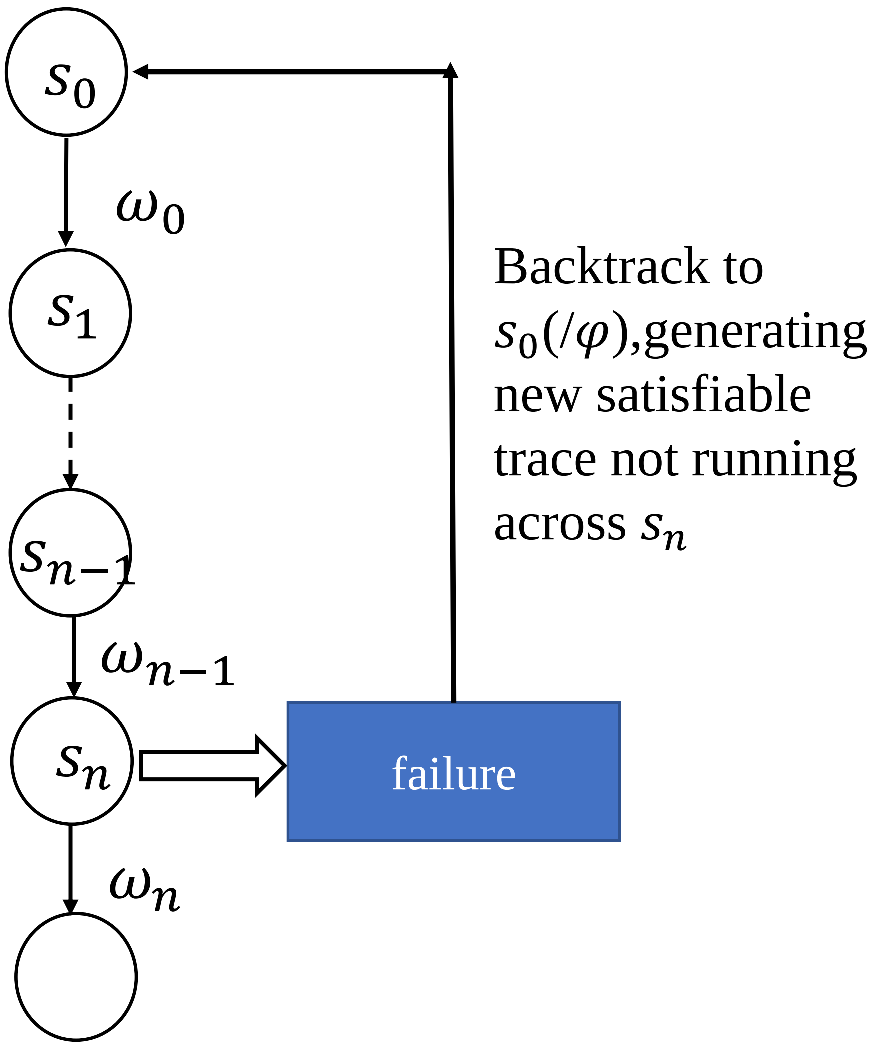


Is there Y∈ $2^{\mathcal{Y}}$ such that
for every X ∈ $2^{\mathcal{X}}$,
the transition (s_n,X ∪ Y, s’)
satisfies
either s’ is winning
or $X \cup Y \models s_n$?



Step 4: Backtrack





Step 5: Termination

- s_0 is winning => realizable

- s_0 cannot find a satisfiable trace => unrealizable

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

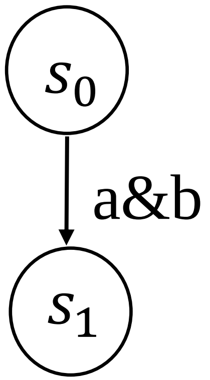
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



find a satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

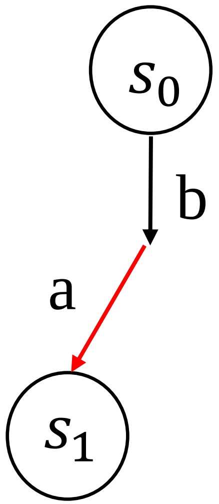
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



select b and check
whether s_0 can be winning.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

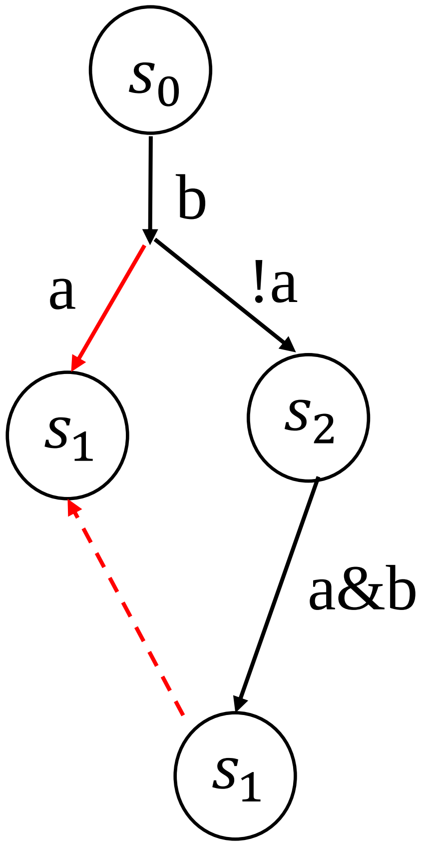
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_0 and fix b, find another satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

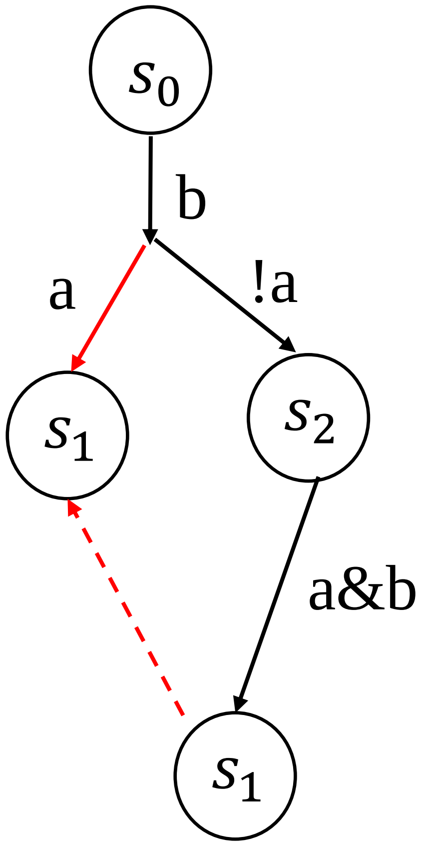
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



Recursively check whether s_2 is winning.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

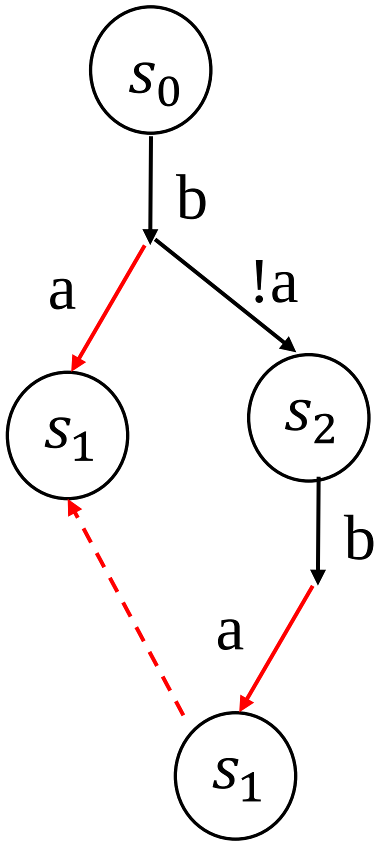
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and fix b, find another satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

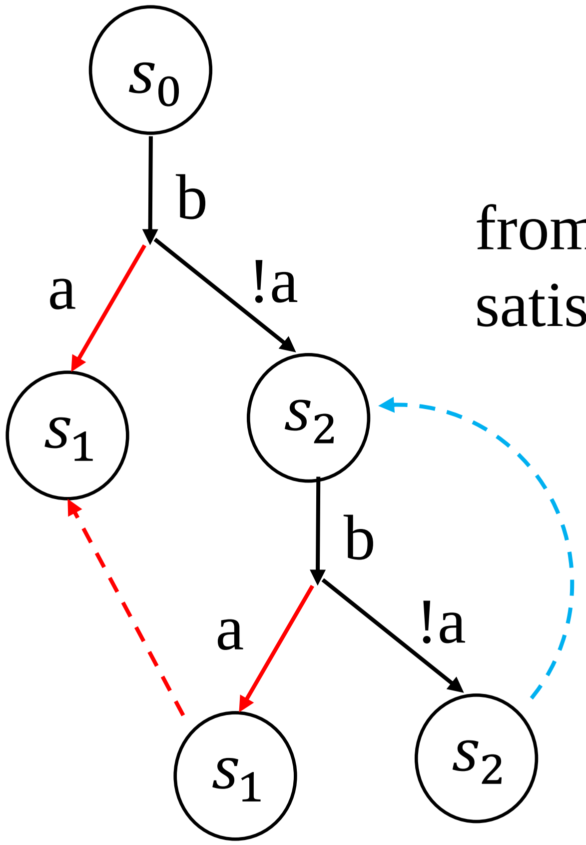
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and fix b, find another satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

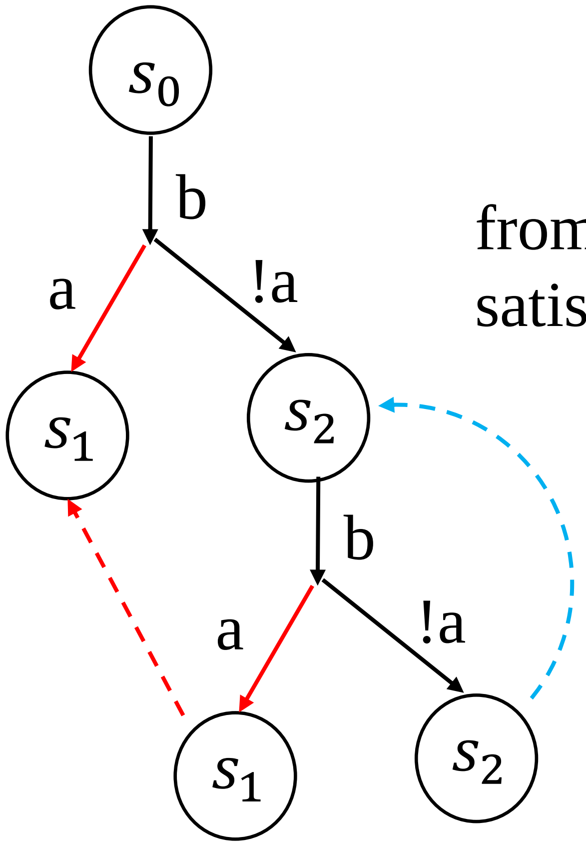
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and fix b, find another satisfiable trace and run.

But the system cannot win in the loop (s_2, b&!a, s_2).

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

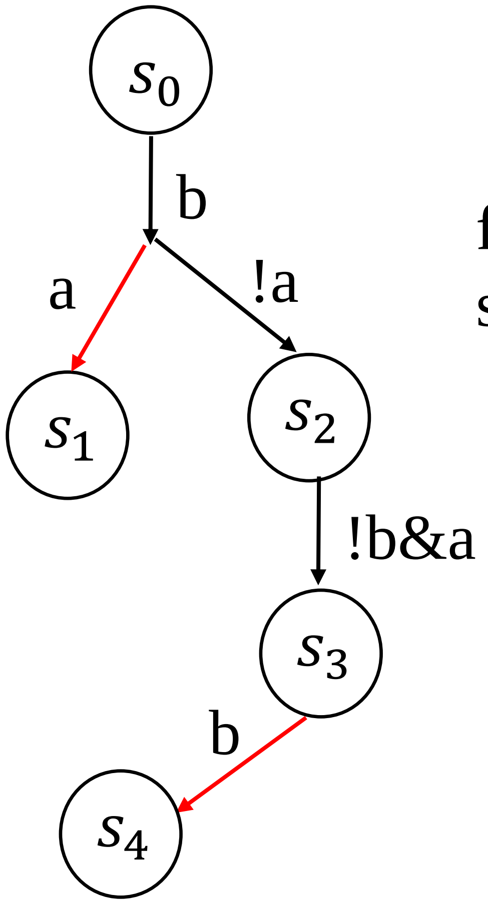
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and block b, find another satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

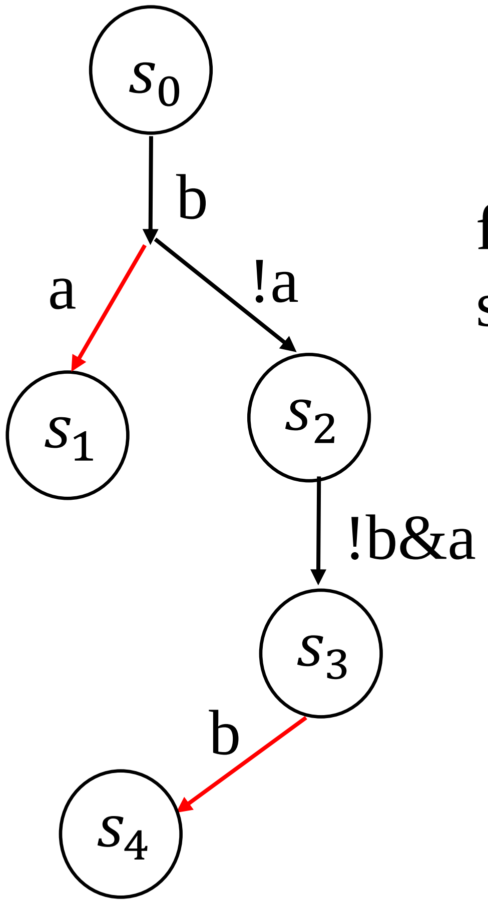
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and block b, find another satisfiable trace and run.

s_3 is winning, so backtrack to s_2.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

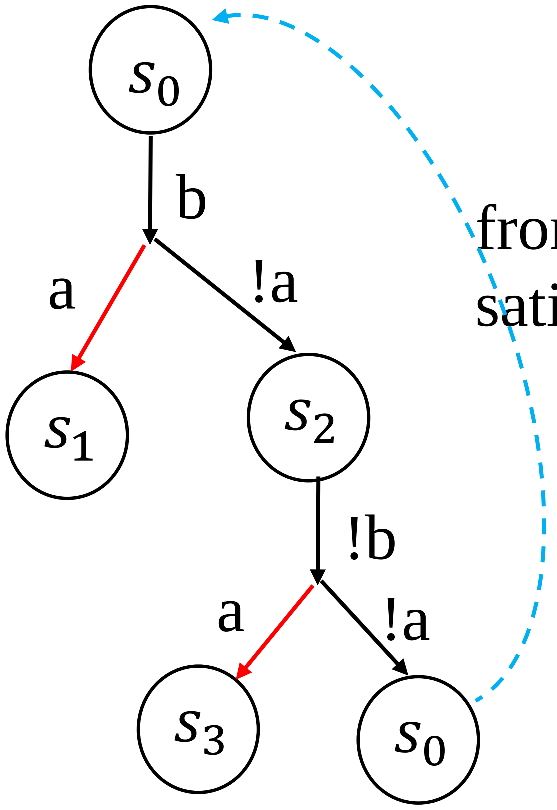
$s_0 = \varphi$

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and fix !b, find another satisfiable trace and run.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

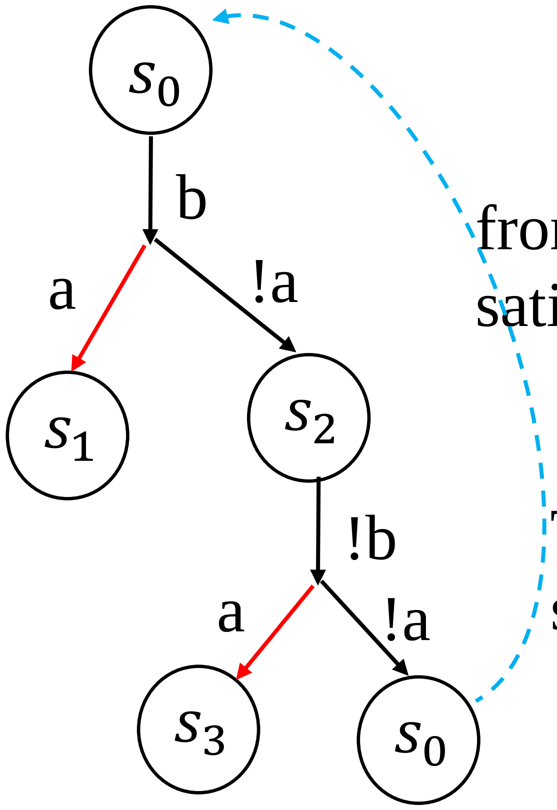
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



from s_2 and fix !b, find another satisfiable trace and run.

The system cannot win in the loop, so s_2 is failure.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

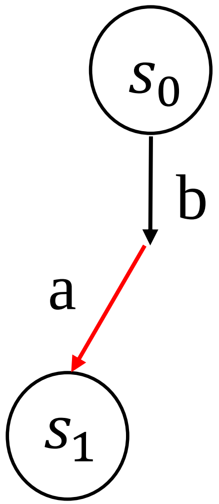
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



From s_0 and select b, we cannot find another satisfiable trace not running across s_2.

The same happens when starting from s_0 and select !b.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

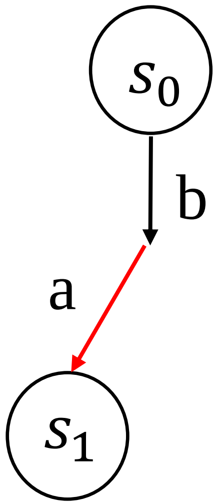
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

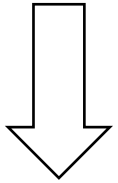
s_3= FG b

s_4= G b



From s_0 and select b, we cannot find another satisfiable trace not running across s_2.

The same happens when starting from s_0 and select !b.



s_0 is a failure state.

Example

- φ = F a & FG b and $\mathcal{X}$ = {a}, $\mathcal{Y}$ = {b}

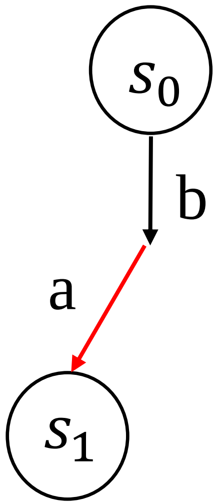
s_0=φ

s_1=G b | FG b

s_2=Fa & (G b | FG b)

s_3= FG b

s_4= G b



From s_0 and select b, we cannot find another satisfiable trace not running across s_2.

The same happens when starting from s_0 and select !b.

s_0 is a failure state.

φ is unrealizable!

Experimental Set-up

- SVS: implemented based on aaltaf [AAAI 2019]
- Cythia: the most recent LTLf synthesis tool [IJICAI 2022]
- OLFS: Our previous LTLf synthesis tool [AAAI 2021]
- Benchmarks: 1494 instances in total, including 40 Pattern instances, 54 Two-player-Games instances and 1400 Random instances

Results

Table 1: Summary of results: pairwise comparison

Comparing	Uniquely solved by SVS	Uniquely solved by 'other'	Solved faster by SVS	Solved faster by 'other'
SVS/ OLFS	246	12	39	134
SVS/ Cynthia	136	32	65	219

Table 2: Summary of results

Tool	Realizable		Unrealizable	
	Solved	Uniquely solved	Solved	Uniquely solved
SVS	189	11	232	107
OLFS	89	1	98	3
Cynthia	189	9	128	15

Summary

- We present a new LTLf synthesis approach by using satisfiability checking
- The experimental results show the promise of the new approach
- In future, we will explore more effective heuristics to continually improve the overall performance

Q&A