Fairness Modulo Theory: A New Approach to LTL Software Model Checking

Daniel Dietsch, Matthias Heizmann, Vincent Langenfeld, and Andreas Podelski

Presented by Henrich Lauko

December 5, 2016

Talk outline

1. Ultimate Automizer algorithm

Talk outline

1. Ultimate Automizer algorithm
2. Reachability solving

Talk outline

1. Ultimate Automizer algorithm
2. Reachability solving
3. Interpolants and infeasibility proofs

Talk outline

1. Ultimate Automizer algorithm
2. Reachability solving
3. Interpolants and infeasibility proofs
4. LTL model checking

Ultimate Automizer

Programs as languages

A program defines a language over program statements.

- alphabet = the set of program statements
- finite automaton = control flow graph
- accepting states = error locations

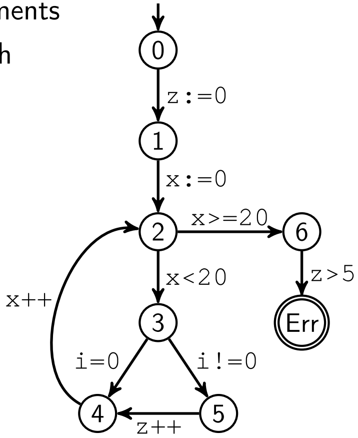
Ultimate Automizer

Programs as languages

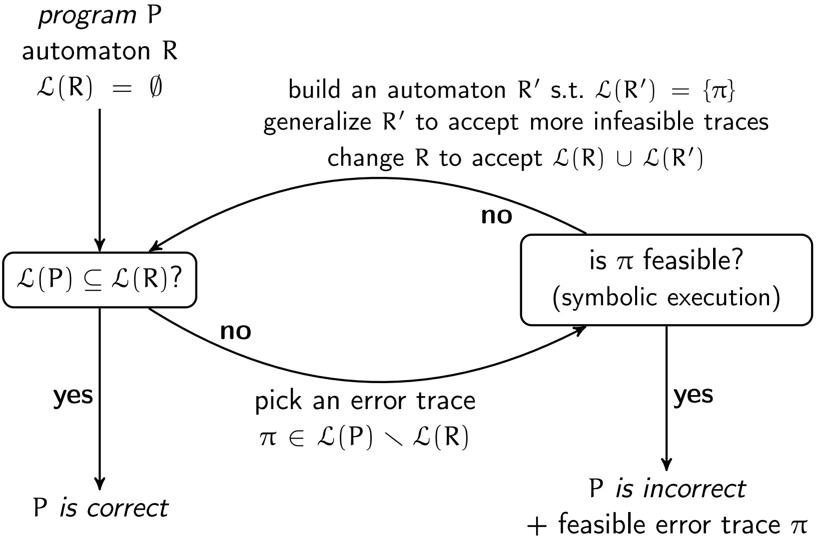
A program defines a language over program statements.

- alphabet = the set of program statements
- finite automaton = control flow graph
- accepting states = error locations

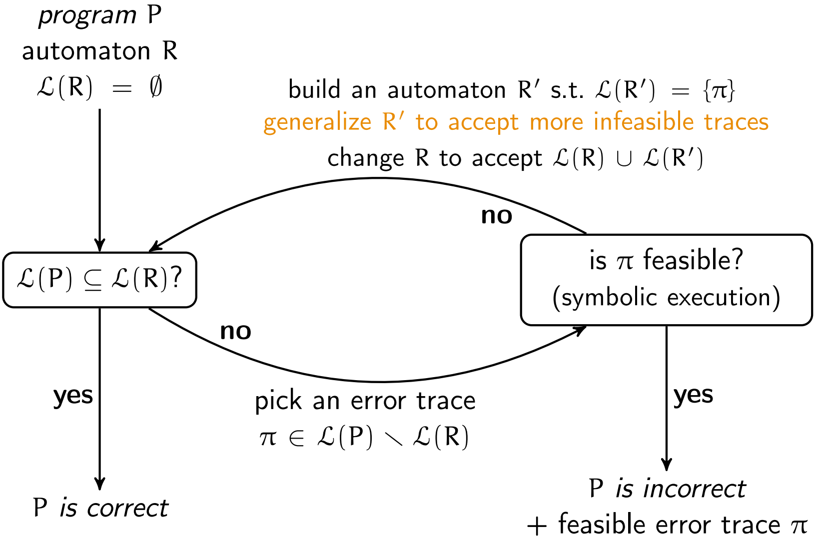
```
int z:=0;
int x:=0;
while(x<20){
     if(i!=0){
          z++;
     }
     x++;
}
assert(z<=5)
```



Automizer algorithm



Automizer algorithm

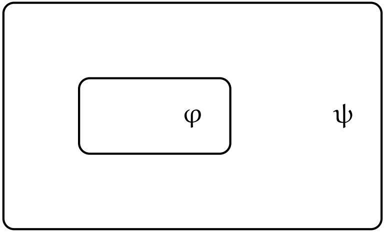


Generalizition of infeasible paths

Craig's interpolants

Let φ, ψ be two first-order formulae such that $\varphi \implies \psi$. Then there exists a first order-formula θ called *interpolant* such that

- all non-logical symbols in θ occur in both φ and ψ,
- $\varphi \implies \theta \implies \psi$.

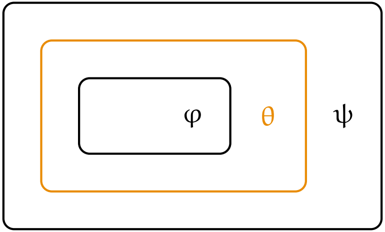


Generalizition of infeasible paths

Craig's interpolants

Let φ, ψ be two first-order formulae such that $\varphi \implies \psi$. Then there exists a first order-formula θ called *interpolant* such that

- all non-logical symbols in θ occur in both φ and ψ,
- $\varphi \implies \theta \implies \psi$.



Generalizition of infeasible paths

Craig's interpolants

Let φ, ψ be two first-order formulae such that $\varphi \implies \psi$. Then there exists a first order-formula θ called *interpolant* such that

- all non-logical symbols in θ occur in both φ and ψ,
- $\varphi \implies \theta \implies \psi$.

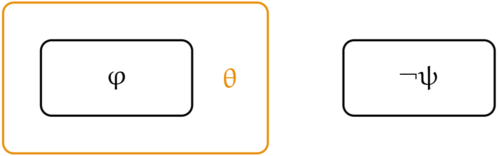
$\boxed{\varphi}$ $\qquad$ $\boxed{\neg\psi}$

Generalizition of infeasible paths

Craig's interpolants

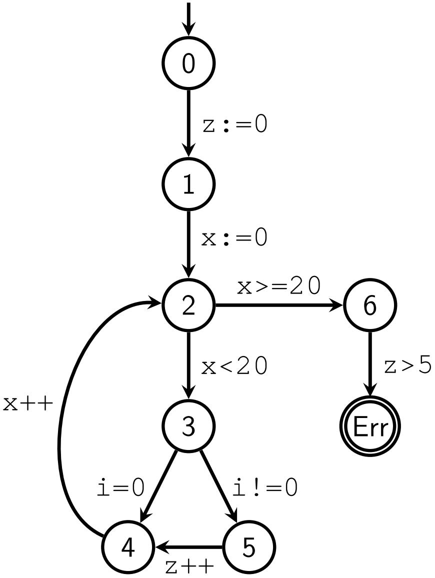
Let φ, ψ be two first-order formulae such that $\varphi \implies \psi$. Then there exists a first order-formula θ called *interpolant* such that

- all non-logical symbols in θ occur in both φ and ψ,
- $\varphi \implies \theta \implies \psi$.



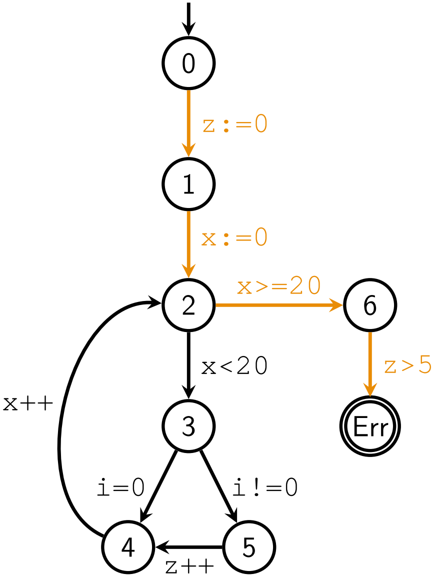
Example: Error path

```
int  z:=0;
int  x:=0;
while(x<20){
     if(i!=0){
          z++;
     }
     x++;
}
assert(z<=5)
```

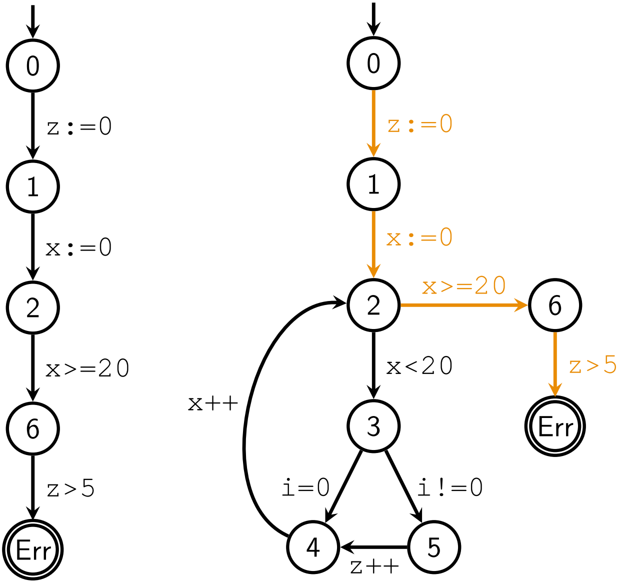


Example: Error path

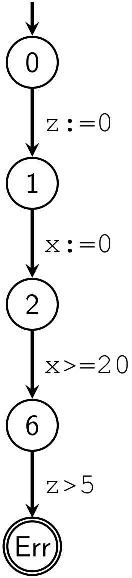
```
int z:=0;
int x:=0;
while(x<20){
    if(i!=0){
        z++;
    }
    x++;
}
assert(z<=5)
```



Example: Error path



Error feasibility analysis

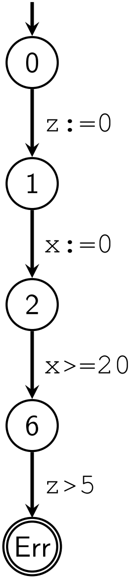


Symbolic execution produces:

$$z = 0 \wedge x = 0 \wedge x \geqslant 20 \wedge z > 5 \equiv \text{false}$$

So the error trace is *infeasible*.

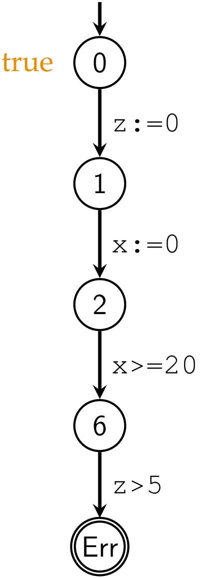
Generalization of infeasible trace



true

$$z = 0 \ \land \ x = 0 \ \land$$
$$\land \ x \geqslant 20 \ \land \ z > 5$$

Generalization of infeasible trace

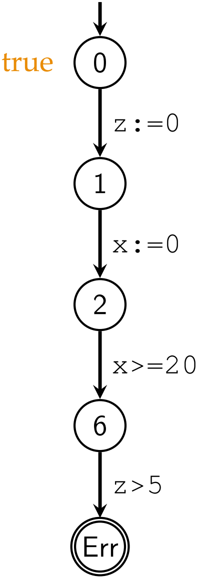


true

true

$$z = 0 \ \wedge \ x = 0 \ \wedge \ \wedge \ x \geqslant 20 \ \wedge \ z > 5$$

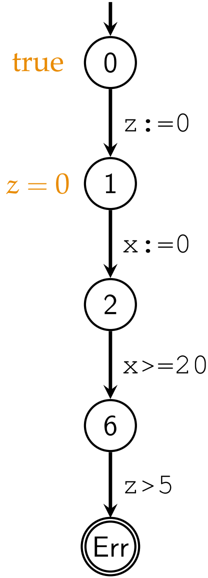
Generalization of infeasible trace



$$\text{true} \ \wedge \ z = 0$$

$$x = 0 \ \wedge$$
$$\wedge \ x \geqslant 20 \ \wedge \ z > 5$$

Generalization of infeasible trace



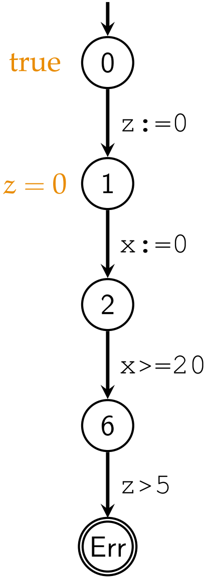
$\text{true} \ \wedge \ z = 0$

$z = 0$

$x = 0 \ \wedge$

$\wedge \ x \geqslant 20 \ \wedge \ z > 5$

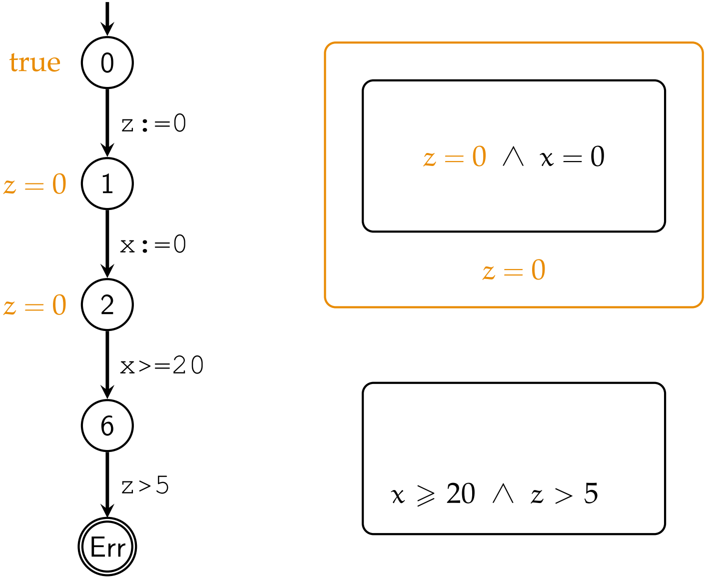
Generalization of infeasible trace



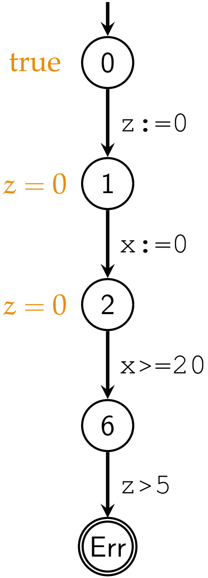
$$z = 0 \;\wedge\; x = 0$$

$$x \geqslant 20 \;\wedge\; z > 5$$

Generalization of infeasible trace



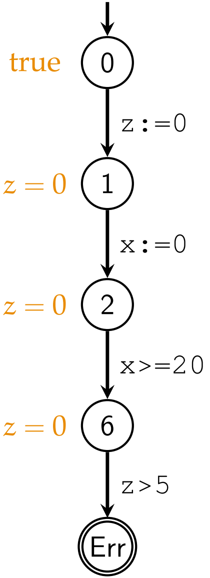
Generalization of infeasible trace



$z = 0 \;\wedge\; x \geqslant 20$

$z \;>\; 5$

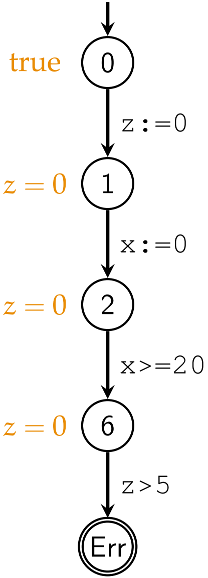
Generalization of infeasible trace



$z = 0 \;\wedge\; x \geqslant 20$

$z = 0$

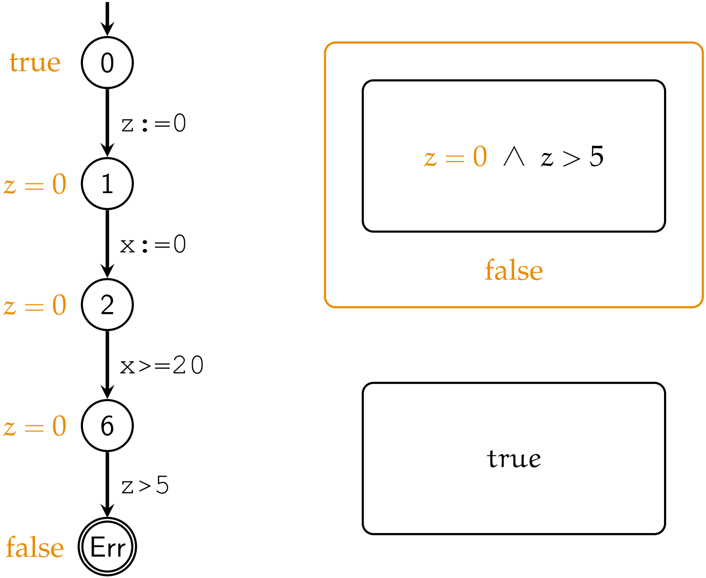
$z > 5$



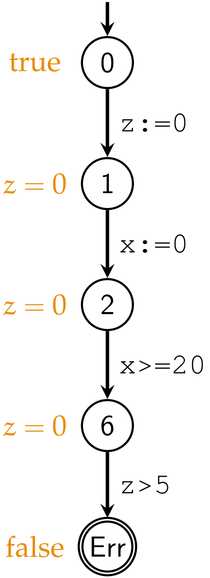
$$z = 0 \ \wedge \ z > 5$$

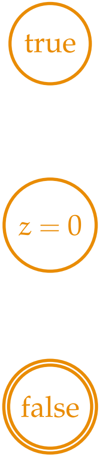
$$\text{true}$$

Generalization of infeasible trace



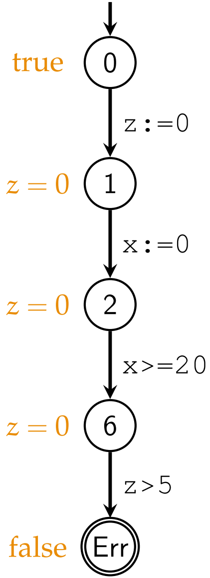
Generalization of infeasible trace

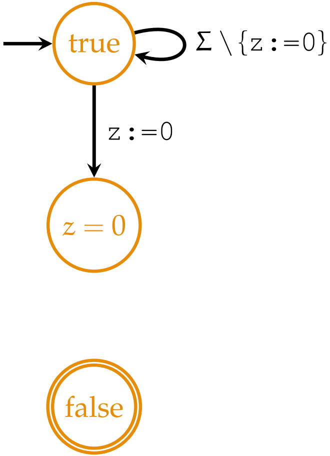




$$\Sigma = \{\texttt{z:=0},\ \texttt{x:=0},\ \texttt{x>=20},\ \texttt{x<20},\ \texttt{i=0},\ \texttt{i!=0},\ \texttt{z++},\ \texttt{x++},\ \texttt{z>5}\}$$

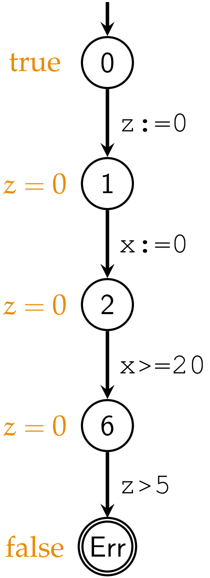
Generalization of infeasible trace

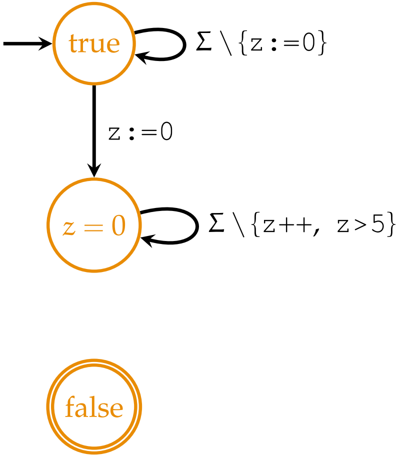




$\Sigma = \{$z:=0, x:=0, x>=20, x<20, i=0, i!=0, z++, x++, z>5$\}$

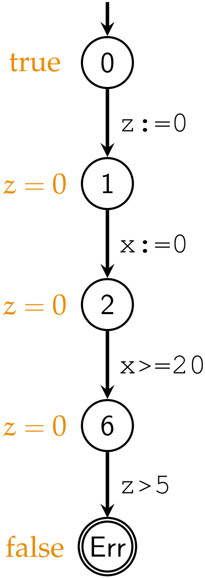
Generalization of infeasible trace

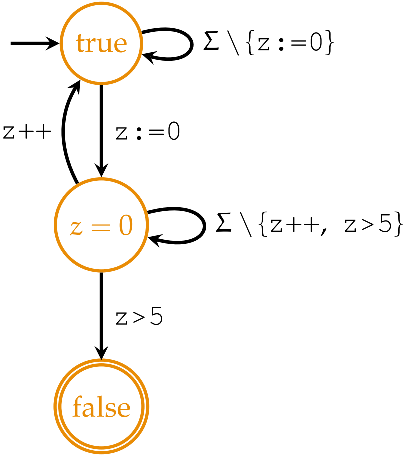




$\Sigma = \{$`z:=0`, `x:=0`, `x>=20`, `x<20`,
`i=0`, `i!=0`, `z++`, `x++`, `z>5`$\}$

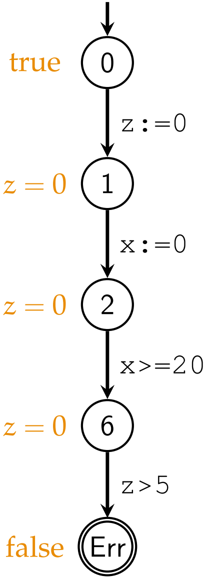
Generalization of infeasible trace

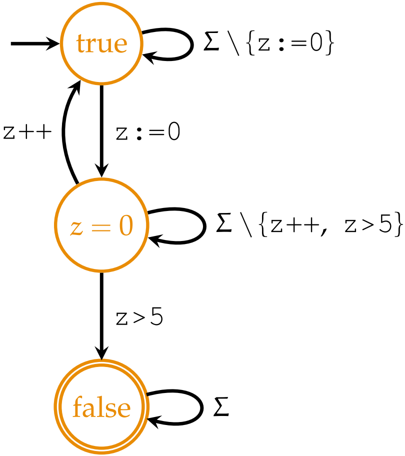




$\Sigma = \{$`z:=0`, `x:=0`, `x>=20`, `x<20`, `i=0`, `i!=0`, `z++`, `x++`, `z>5`$\}$

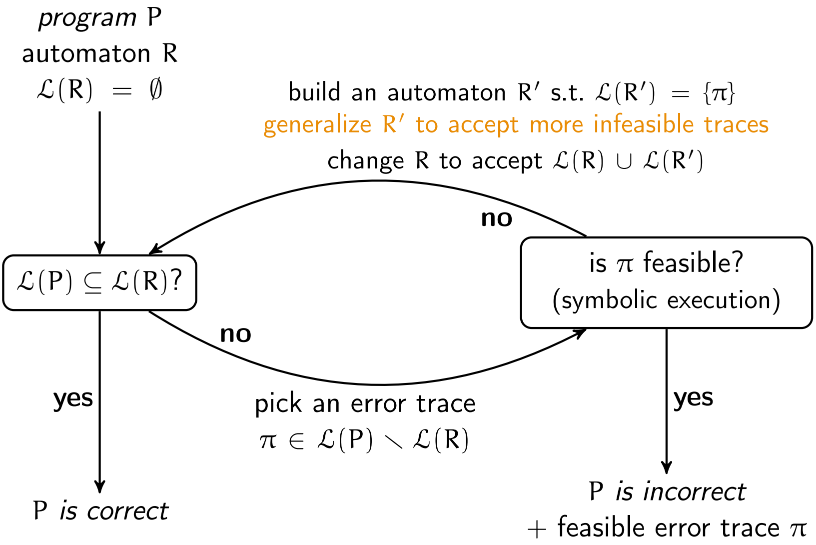
Generalization of infeasible trace



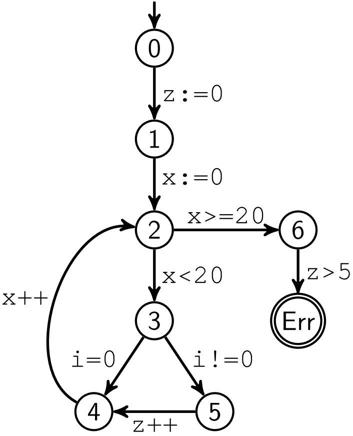


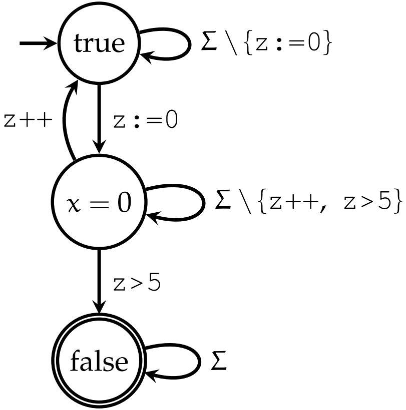
$\Sigma = \{$`z:=0`, `x:=0`, `x>=20`, `x<20`,
`i=0`, `i!=0`, `z++`, `x++`, `z>5`$\}$

Automizer algorithm

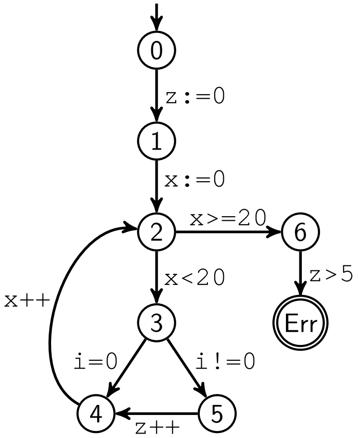


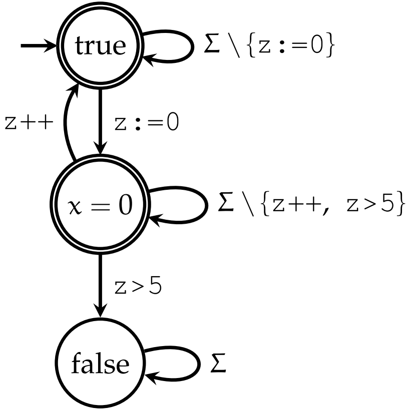
$\mathcal{L}(P) \subseteq \mathcal{L}(R)$?





$\mathcal{L}(P) \subseteq \mathcal{L}(R)$?





$\mathcal{L}(P) \subseteq \mathcal{L}(R)$ iff the language of the synchronous product of P and complemented R is empty.

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

Automizer approach:

- LTL property φ is represented by Büchi automaton $\mathcal{A}_\varphi$

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

Automizer approach:

- LTL property φ is represented by Büchi automaton $\mathcal{A}_\varphi$
- creates Büchi program $\mathcal{B} = \mathcal{P} \times \mathcal{A}_{\neg\varphi}$

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

Automizer approach:

- LTL property φ is represented by Büchi automaton $\mathcal{A}_\varphi$
- creates Büchi program $\mathcal{B} = \mathcal{P} \times \mathcal{A}_{\neg\varphi}$
- violation of property is proven via a feasible *fair* path

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

Automizer approach:

- LTL property φ is represented by Büchi automaton $\mathcal{A}_\varphi$
- creates Büchi program $\mathcal{B} = \mathcal{P} \times \mathcal{A}_{\neg\varphi}$
- violation of property is proven via a feasible *fair* path

LTL model checking

LTL model checking problem

Does the program $\mathcal{P}$ *satisfy* the property φ?

Automizer approach:

- LTL property φ is represented by Büchi automaton $\mathcal{A}_\varphi$
- creates Büchi program $\mathcal{B} = \mathcal{P} \times \mathcal{A}_{\neg\varphi}$
- violation of property is proven via a feasible *fair* path

Fair path

Path is fair if it visits set of accepting states infinitely often.

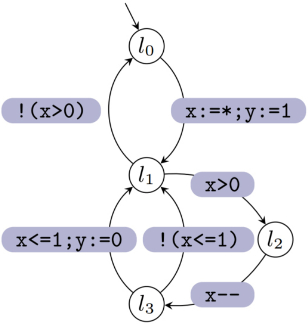
Example: Büchi program

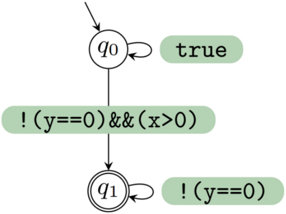
Program:

```
int x, y;
while( true ){
    x := *;
    y := 1;
    while( x > 0 ){
        x--;
        if( x <= 1 )
            y := 0;
    }
}
```

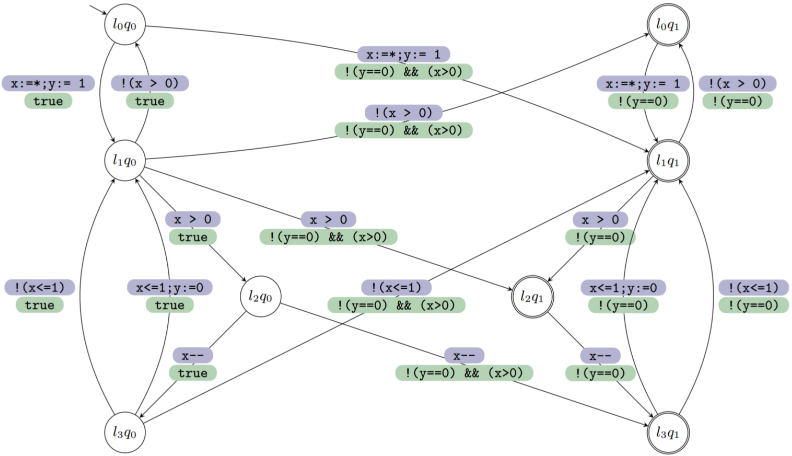
LTL property:

$$\varphi \equiv G(x > 0 \implies F(y = 0))$$





Example: Büchi program



Infeasible path

An infinite path is infeasible if:

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

$\boxed{x := y}$ $\boxed{x > y}$ $\boxed{x--}$ $\boxed{x > y}$ $\boxed{x--}$ $\boxed{x > y}$ $\ldots$

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

   $\boxed{\mathrm{x} := \mathrm{y}}$ $\boxed{\mathrm{x} > \mathrm{y}}$ $\boxed{\mathrm{x} - -}$ $\boxed{\mathrm{x} > \mathrm{y}}$ $\boxed{\mathrm{x} - -}$ $\boxed{\mathrm{x} > \mathrm{y}}$ $\dots$

2. there is a ranking function for the path, ie. the execution is terminating.

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

   $\boxed{x := y}\ \boxed{x > y}\ \boxed{x--}\ \boxed{x > y}\ \boxed{x--}\ \boxed{x > y}\ \ldots$

2. there is a ranking function for the path, ie. the execution is terminating.

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

   $\boxed{x := y}\ \boxed{x > y}\ \boxed{x - -}\ \boxed{x > y}\ \boxed{x - -}\ \boxed{x > y}\ \ldots$

2. there is a ranking function for the path, ie. the execution is terminating.

   $\boxed{x - -}\ \boxed{x > 0}\ \boxed{x - -}\ \boxed{x > 0}\ \boxed{x - -}\ \boxed{x > 0}\ \ldots$

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

   $\boxed{x := y}$ $\boxed{x > y}$ $\boxed{x--}$ $\boxed{x > y}$ $\boxed{x--}$ $\boxed{x > y}$ ...

2. there is a ranking function for the path, ie. the execution is terminating.

   $\boxed{x--}$ $\boxed{x > 0}$ $\boxed{x--}$ $\boxed{x > 0}$ $\boxed{x--}$ $\boxed{x > 0}$ ...

   Termination proof: $r(x, y) = x$

Infeasible path

An infinite path is infeasible if:

1. some finite prefix is not feasible.

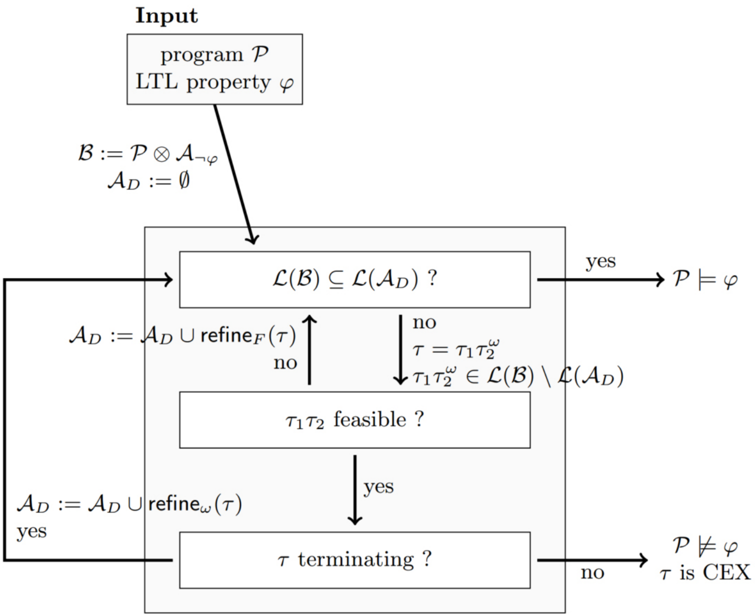
   $\boxed{x := y}\ \boxed{x > y}\ \boxed{x--}\ \boxed{x > y}\ \boxed{x--}\ \boxed{x > y}\ \ldots$

2. there is a ranking function for the path, ie. the execution is terminating.

   $\boxed{x--}\ \boxed{x > 0}\ \boxed{x--}\ \boxed{x > 0}\ \boxed{x--}\ \boxed{x > 0}\ \ldots$

   Termination proof: $r(x, y) = x$

Ultimate LTL Automizer



Implementation details

- using parser for ANSI C

Implementation details

- using parser for ANSI C
- LTL property in ACLS format

Implementation details

- using parser for ANSI C
- LTL property in ACLS format
- using LTL2BA

Implementation details

- using parser for ANSI C
- LTL property in ACLS format
- using LTL2BA
- own source-to-source transformations

Implementation details

- using parser for ANSI C
- LTL property in ACLS format
- using LTL2BA
- own source-to-source transformations
- own trace abstraction, ranking function synthesis, automata manipulations

Some benchmarks

Program	Lines	φ	Term. [21] Time (s)	Term. [21] Result	DP [23] Time (s)	DP [23] Result	ULTIMATE LTLAUTOMIZER Time (s)	ULTIMATE LTLAUTOMIZER Result	$\lvert\mathbf{r}_F\rvert$	$\lvert\mathbf{r}_\omega\rvert$	Inc. (%)
Ex. Sec. 2 of [23]	5	$\Diamond\Box p$	2.32	✔	1.98	✔	0.51	✔	1	0	122
Ex. Fig. 8 of [21]	34	$\Box(p \Rightarrow \Diamond q)$	209.64	✔	27.94	✔	0.72	✔	2	0	186
Toy acquire/release	14	$\Box(p \Rightarrow \Diamond q)$	103.48	✔	14.18	✔	0.44	✔	1	1	129
Toy linear arith. 1	13	$p \Rightarrow \Diamond q$	126.86	(✔)	34.51	(✔)	1.10	✗	5	1	0.28
Toy linear arith. 2	13	$p \Rightarrow \Diamond q$	**T.O.**	**T.O.**	6.74	✔	0.82	✔	4	2	0.24
PostgreSQL strmsrv	259	$\Box(p \Rightarrow \Diamond\Box q)$	**T.O.**	**T.O.**	9.56	✔	1.04	✔	2	0	216
PostgreSQL strmsrv + bug	259	$\Box(p \Rightarrow \Diamond\Box q)$	87.31	(✗)	47.16	(✗)	0.66	✔	2	0	216
PostgreSQL pgarch	61	$\Diamond\Box p$	31.50	(✔)	15.20	(✔)	0.33	✗	2	0	209
PostgreSQL dropbuf	152	$\Box p$	**T.O.**	**T.O.**	1.14	(✔)	3.57	✗	1	1	148
PostgreSQL dropbuf	152	$\Box(p \Rightarrow \Diamond q)$	53.99	✔	27.54	✔	1.37	✔	2	1	168
Apache `accept()`	314	$\Box p \Rightarrow \Box\Diamond q$	**T.O.**	**T.O.**	197.41	✔	502.15	**OOM**	-	-	209
Apache progress	314	$\Box(p \Rightarrow (\Diamond q_1 \vee \Diamond q_2))$	685.34	✔	684.24	✔	2.01	✔	4	0	209
Windows OS 1	180	$\Box(p \Rightarrow \Diamond q)$	901.81	✔	539.00	✔	43.59	✔	1	1	178
Windows OS 2	158	$\Diamond\Box p$	16.47	✔	52.10	✔	0.11	✔	1	0	176
Windows OS 2 + bug	158	$\Diamond\Box p$	26.15	✗	30.37	✗	0.22	✗	1	0	174
Windows OS 3	14	$\Diamond\Box p$	4.21	✔	15.75	✔	0.08	✔	2	0	220
Windows OS 4	327	$\Box(p \Rightarrow \Diamond q)$	**T.O.**	**T.O.**	1,114.18	✔	1.86	✔	1	3	207
Windows OS 4	327	$(\Diamond p) \vee (\Diamond q)$	1,223.96	✔	100.68	✔	-	**N.R.**	-	-	-
Windows OS 5	648	$\Box(p \Rightarrow \Diamond q)$	**T.O.**	**T.O.**	**T.O.**	**T.O.**	20.76	✔	1	16	190
Windows OS 6	13	$\Diamond\Box p$	149.41	✔	59.56	✔	**T.O.**	**T.O.**	6	8	158
Windows OS 6 + bug	13	$\Diamond\Box p$	6.06	✗	22.12	✗	0.05	✗	0	0	61
Windows OS 7	13	$\Box\Diamond p$	**T.O.**	**T.O.**	55.77	✔	0.91	✔	2	11	161
Windows OS 8	181	$\Diamond\Box p$	**T.O.**	**T.O.**	5.24	✔	53.55	✔	4	55	168

Some benchmarks

Program set	Avg. Lines	|Set|	✔	✗	**T.O.**	**OOM**	★ (**N.R.**)	Statistics for ✔ and ✗: Avg. Time (s)	Avg. $\lvert \mathbf{r}_F \rvert$	Avg. $\lvert \mathbf{r}_\omega \rvert$	Inc. (%)
RERS P14	514	50	19	21	2	0	8	107.21	21	< 1	329
RERS P15	1353	50	24	0	11	12	3	103.46	17	< 1	369
RERS P16	1304	50	15	1	16	14	4	297.34	32	< 1	362
RERS P17	2100	50	26	0	9	9	6	56.38	12	< 1	324
RERS P18	3306	50	21	0	17	10	2	262.03	24	< 1	297
RERS P19	8079	50	0	0	28	17	5	-	-	-	-
coolant	65	18	6	10	2	0	0	1.75	2	1	258
Benchmarks from Tab. 1	157	23	15	5	1	1	0 (1)	16.78	2	5	184