First-Order Transitive Closure Axiomatization via Iterative Invariant Injections

Aboubakr Achraf El Ghazi, Mana Taghdiri, Mihai Herda

Karlsruhe Institute of Technology, Germany

NFM 2015
Pasadena, April 27

Motivation

- Relational logics are well suited for modelling linked data structures.
- Transitive closure (TC) plays a distinguished role.
  - All instances of `Tree` are acyclic:

$$\forall t : Tree.\ t \notin t \,.\, (left \cup right)^{+}$$

  - The data d occurs at most once in the tree t:

$$\#((t \,.\, (left \cup right)^{*}) \cap (data \,.\, d)) \leq 1$$

Motivation

- Relational logics are well suited for modelling linked data structures.
- Transitive closure (TC) plays a distinguished role.
  - All instances of `Tree` are acyclic:

$$\forall t : \mathit{Tree}.\ t \notin t \,.\, (\mathit{left} \cup \mathit{right})^{+}$$

  - The data d occurs at most once in the tree t:

$$\#((t \,.\, (\mathit{left} \cup \mathit{right})^{*}) \cap (\mathit{data} \,.\, d)) \leq 1$$

Motivation

- Relational logics are well suited for modelling linked data structures.
- Transitive closure (TC) plays a distinguished role.
  - All instances of `Tree` are acyclic:

$$\forall t : Tree.\ t \notin t \,.\, (left \cup right)^+$$

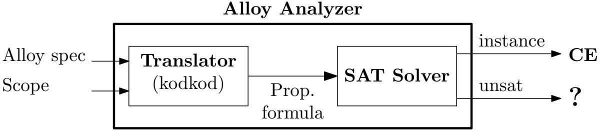
  - The data d occurs at most once in the tree t:

$$\#((t \,.\, (left \cup right)^*) \cap (data \,.\, d)) \leq 1$$

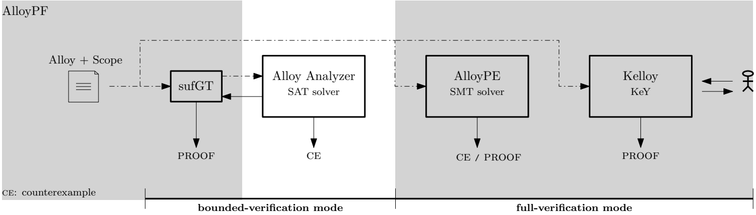
Motivation

Alloy – a typed first-order relational language.

- Alloy Analyzer – *bounded model finder*



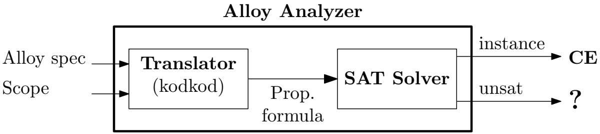
- AlloyPF – *our framework for proving Alloy specifications*



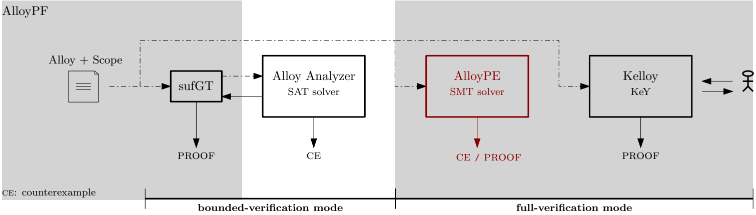
Motivation

Alloy – a typed first-order relational language.

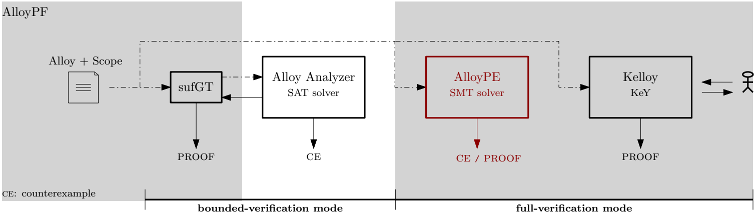
- Alloy Analyzer – *bounded model finder*



- AlloyPF – *our framework for proving Alloy specifications*



Motivation



- RFOL – *Pure* first-order encoding of relational operators
  - The exceptions are: TC, set cardinality and ordering
  - For these operators, the integer theory is involved
- SMT based reasoning – *full automatic*

Motivation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	TO	–
	soundness2	8	28.84	TO	–
	completeness	7	32.52	TO	–
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sound CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE

Table : Experiment results of FM'2011

Motivation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	TO	–
	soundness2	8	28.84	TO	–
	completeness	7	32.52	TO	–
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sound CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE

Table : Experiment results of FM'2011

no transitive closure occurrence

Motivation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	TO	–
	soundness2	8	28.84	TO	–
	completeness	7	32.52	TO	–
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sound CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE

Table : Experiment results of FM'2011

transitive closure occurrence!

Research Questions

- Is there a logic fragment for the integer based axiomatization?
- Is the integer theory really necessary for this fragment?
- What kind of axiomatization can help beyond this fragment?

WTC Axiomatization

- (Usual) Integer based TC Axiomatization

  $\forall x_1, x_2.\ (x_1, x_2) \in tc_R \leftrightarrow \exists n.\ (x_1, x_2) \in R^n$

- Weak *first-order* TC axiomatization (WTC)

  $\forall x_1, x_2.\ (x_1, x_2) \in R \rightarrow (x_1, x_2) \in tc_R$
  $\forall x_1, x_2, x_3.\ (x_1, x_2) \in tc_R \wedge (x_2, x_3) \in tc_R \rightarrow (x_1, x_3) \in tc_R$

WTC Complete Fragment – w.r.t. R-Paths

F^{CNF}: $\dots$
$\dots \vee (a_1, b_1) \notin tc_R \vee \dots$
$\dots$
$\dots \vee (a_2, b_2) \in tc_R \vee \dots$
$\dots$
$\dots \vee (a_2, b_2) \notin tc_R \vee \dots$
$\dots$

WTC: $R \subseteq tc_R$
$Transitive(tc_R)$
$\dots$

Theorem (WTC complete fragment)

Let F be a first-order relational formula, R a binary relations, and u a tuple where the R-path (literal) $u \in tc_R$ occurs only negative in F^{CNF}. Then, F is unsatisfiable modulo $\mathcal{T}^{R^+}_{tc_R|u}$ iff it is unsatisfiable modulo $\mathcal{T}^{WTC}_{tc_R|u}$.

WTC Complete Fragment – w.r.t. R-Paths

$$
\begin{aligned}
F^{CNF}: \quad & \dots \\
& \dots \vee (a_1, b_1) \notin tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \notin tc_R \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Theorem (WTC complete fragment)

Let F be a first-order relational formula, R a binary relations, and u a tuple where the R-path (literal) $u \in tc_R$ occurs only negative in F^{CNF}. Then, F is unsatisfiable modulo $\mathcal{T}^{R^+}_{tc_R|u}$ iff it is unsatisfiable modulo $\mathcal{T}^{WTC}_{tc_R|u}$.

WTC Complete Fragment – w.r.t. R-Paths

$$
\begin{aligned}
F^{CNF}: \quad & \dots \\
& \dots \vee (a_1, b_1) \notin tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \notin tc_R \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Theorem (WTC complete fragment)

Let F be a first-order relational formula, R a binary relations, and u a tuple where the R-path (literal) $u \in tc_R$ occurs only negative in F^{CNF}. Then, F is unsatisfiable modulo $\mathcal{T}^{R^+}_{tc_R|u}$ iff it is unsatisfiable modulo $\mathcal{T}^{WTC}_{tc_R|u}$.

WTC Complete Fragment – w.r.t. R-Paths

$$
\begin{aligned}
F^{CNF}: \quad & \dots \\
& \dots \vee (a_1, b_1) \notin tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \notin tc_R \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Theorem (WTC complete fragment)

Let F be a first-order relational formula, R a binary relations, and u a tuple where the R-path (literal) $u \in tc_R$ occurs only negative in F^{CNF}. Then, F is unsatisfiable modulo $\mathcal{T}^{R^+}_{tc_R|u}$ iff it is unsatisfiable modulo $\mathcal{T}^{WTC}_{tc_R|u}$.

$$\mathcal{T}^{R^+}_{tc_R|u}: \quad \{\mathcal{M} = (\bar{M}, M) \mid M(tc_R)|_{M(u)} = M(R)^+|_{M(u)}\}$$

WTC Complete Fragment – w.r.t. *R*-Paths

$$
\begin{aligned}
F^{CNF}: \quad & \dots \\
& \dots \vee (a_1, b_1) \notin tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \notin tc_R \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Theorem (WTC complete fragment)

Let F be a first-order relational formula, R a binary relations, and u a tuple where the R-path (literal) $u \in tc_R$ occurs only negative in F^{CNF}. Then, F is unsatisfiable modulo $\mathcal{T}^{R^+}_{tc_R|u}$ iff it is unsatisfiable modulo $\mathcal{T}^{WTC}_{tc_R|u}$.

$$\mathcal{T}^{WTC}_{tc_R|u}: \quad \{\mathcal{M} = (\bar{M}, M) \mid M(tc_R)|_{M(u)} = M(R)^{WTC}|_{M(u)}\}$$

Motivation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	TO	–
	soundness2	8	28.84	TO	–
	completeness	7	32.52	TO	–
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sound CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE

positive R-paths!

Semantical Extension of WTC Fragment

- WTC-Theorem
  - Provides a syntactical def. of *safe* R-paths
  - Provides a syntactical def. of *unsafe* R-paths (UP)
  - However, unsafe is not precise enough
- *Essential* R-paths – *semantical extension of unsafe*

  Definition

  Given a binary relation R and a refutable first-order relational formula F modulo $\mathcal{T}_{tc_R}^{R^+}$, then an unsafe R-path $p \in UP$ is *essential* if there exists a model $\mathcal{M}$ of F where

$$\forall p' \in UP \setminus \{p\}.\ \mathcal{M}(tc_R)|_{\mathcal{M}(p'_u)} = M(R)^+|_{\mathcal{M}(p'_u)} \text{ and}$$
$$\mathcal{M}(tc_R)|_{\mathcal{M}(p_u)} = M(R)^{WTC}|_{\mathcal{M}(p_u)}$$

- Use of R^+ makes it impractical – *heuristics are needed*

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F: \quad & \dots \\
& \dots \lor (a_1, b_1) \in tc_R \lor \dots \\
& \dots \\
& \dots \lor (a, b) \in tc_R \lor \dots \\
& \dots \\
& \dots \lor (a_2, b_2) \in tc_R \lor \dots \\
& \dots \\
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F: \quad & \ldots \\
& \cdots \vee (a_1, b_1) \in tc_R \vee \ldots \\
& \ldots \\
& \cdots \vee (a, b) \in tc_R \vee \ldots \\
& \ldots \\
& \cdots \vee (a_2, b_2) \in tc_R \vee \ldots \\
& \ldots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \ldots
\end{aligned}
$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F|^1_{(a,b)}: \quad & \dots \\
& \dots \vee (a_1, b_1) \in R \vee \dots \\
& \dots \\
& \dots \vee (a, b) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in R \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F|^{n}_{(a,b)}: \quad & \dots \\
& \dots \vee (a_1, b_1) \in \bigcup_{1 \le i \le n} R^i \vee \dots \\
& \dots \\
& \dots \vee (a, b) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in \bigcup_{1 \le i \le n} R^i \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC: \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F|^{n}_{(a,b)} : \quad & \dots \\
& \dots \vee (a_1, b_1) \in \bigcup_{1 \leq i \leq n} R^i \vee \dots \\
& \dots \\
& \dots \vee (a, b) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in \bigcup_{1 \leq i \leq n} R^i \vee \dots \\
& \dots \\
WTC : \quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

$$\forall n. \bigcup_{1 \leq i \leq n} R^i \subseteq R^+$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F|^{\infty}_{(a,b)} :\quad & \dots \\
& \dots \vee (a_1, b_1) \in R^+ \vee \dots \\
& \dots \\
& \dots \vee (a, b) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in R^+ \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC :\quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

$$\forall n. \bigcup_{1 \le i \le n} R^i \subseteq R^+$$

$$\mathcal{M} \models F|^{n}_{(a,b)} \rightarrow \mathcal{M} \models F|^{\infty}_{(a,b)}$$

Essential R-Paths – *Bounded Isolation*

Assuming F is refutable

$$
\begin{aligned}
F|^{\infty}_{(a,b)} :\quad & \dots \\
& \dots \vee (a_1, b_1) \in R^+ \vee \dots \\
& \dots \\
& \dots \vee (a, b) \in tc_R \vee \dots \\
& \dots \\
& \dots \vee (a_2, b_2) \in R^+ \vee \dots \\
& \dots
\end{aligned}
$$

$$
\begin{aligned}
WTC :\quad & R \subseteq tc_R \\
& Transitive(tc_R) \\
& \dots
\end{aligned}
$$

$$\forall n. \bigcup_{1 \le i \le n} R^i \subseteq R^+$$

$$\mathcal{M} \models F|^{n}_{(a,b)} \rightarrow \mathcal{M} \models F|^{\infty}_{(a,b)}$$

$$F|^{n}_{(a,b)} \overset{if}{=} \begin{cases} SAT & \text{then } p \text{ is essential} \\ else & \text{then } p \text{ is not essential with conf. } n \end{cases}$$

Semantical Extension of WTC Fragment

Definition (bounded R-path isolation)

Let R be a binary relation, F a first-order relational formula, p a difficult R-path in F and n a positive natural number. Then, the n confident isolation of p in F is

$$F|_p^n := F\left[u \in \bigcup_{1 \leq i \leq n} R^i \;/\; u \in tc_R \mid (u \in tc_R) \in UP \setminus \{p\}\right].$$

- If $F|_p^n$ is satisfiable then p is essential,
- otherwise p is not essential (with confidence n).
- As long as $F|_p^n$ is satisfiable further handling of p is needed, otherwise not.

Motivation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	TO	–
	soundness2	8	28.84	TO	–
	completeness	7	32.52	TO	–
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sound CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE

no *essential* *R*-paths

Example

Types : *Heap*, *Obj*
Consts : $h_0, \dots, h_3$: *Heap*, *root*, *live* : *Obj*

Rels : $ref \subseteq Heap \times Obj \times Obj$, $mark \subseteq: Heap \times Obj$
Funs : $tc_{H.ref} : Heap \rightarrow Obj \times Obj$

F :
(1) $h_1 . mark = \emptyset$
(2) $h_0 . ref \subseteq h_1 . ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$
(4) $h_1 . ref \subseteq h_2 . ref$
(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:
(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

Types : *Heap*, *Obj*

Consts : $h_0, \ldots, h_3$: *Heap*, *root*, *live* : *Obj*

Rels : $ref \subseteq Heap \times Obj \times Obj$, $mark \subseteq: Heap \times Obj$

Funs : $tc_{H.ref} : Heap \to Obj \times Obj$

F :

(1) $h_1 \,.\, mark = \emptyset$

(2) $h_0 \,.\, ref \subseteq h_1 \,.\, ref$

(3) $\forall n.\neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 \,.\, mark$

(4) $h_1 \,.\, ref \subseteq h_2 \,.\, ref$

(5) $\forall n.\neg(n \notin h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = \emptyset$

(6) $\forall n.\neg(n \in h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = n \,.\, (h_2 \,.\, ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live \,.\, (h_0 \,.\, ref) \not\subseteq live \,.\, (h_3 \,.\, ref)$

WTC:

(9) $\forall h.\ h \,.\, ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

Types : *Heap*, *Obj*

Consts : $h_0, \ldots, h_3$: *Heap*, *root*, *live* : *Obj*

Rels : $ref \subseteq Heap \times Obj \times Obj$, $mark \subseteq: Heap \times Obj$

Funs : $tc_{H.ref} : Heap \to Obj \times Obj$

F :

(1) $h_1 \,.\, mark = \emptyset$

(2) $h_0 \,.\, ref \subseteq h_1 \,.\, ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 \,.\, mark$

(4) $h_1 \,.\, ref \subseteq h_2 \,.\, ref$

(5) $\forall n. \neg(n \notin h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = n \,.\, (h_2 \,.\, ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live \,.\, (h_0 \,.\, ref) \not\subseteq live \,.\, (h_3 \,.\, ref)$

WTC:

(9) $\forall h.\ h \,.\, ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

$Types : Heap, Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$

$Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Funs : tc_{H.ref} : Heap \to Obj \times Obj$

F :

(1) $h_1 . mark = \emptyset$

(2) $h_0 . ref \subseteq h_1 . ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$

(4) $h_1 . ref \subseteq h_2 . ref$

(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

$Types : Heap, Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$

$Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Funs : tc_{H.ref} : Heap \to Obj \times Obj$

$F :$

(1) $h_1 \,.\, mark = \emptyset$
(2) $h_0 \,.\, ref \subseteq h_1 \,.\, ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \lor n \in h_2 \,.\, mark$
(4) $h_1 \,.\, ref \subseteq h_2 \,.\, ref$
(5) $\forall n. \neg(n \notin h_2 \,.\, mark) \lor n \,.\, (h_3 \,.\, ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 \,.\, mark) \lor n \,.\, (h_3 \,.\, ref) = n \,.\, (h_2 \,.\, ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live \,.\, (h_0 \,.\, ref) \not\subseteq live \,.\, (h_3 \,.\, ref)$

WTC:

(9) $\forall h.\ h \,.\, ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

$Types : Heap, Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$

$Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Funs : tc_{H.ref} : Heap \to Obj \times Obj$

$F :$

(1) $h_1 \,.\, mark = \emptyset$
(2) $h_0 \,.\, ref \subseteq h_1 \,.\, ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 \,.\, mark$
(4) $h_1 \,.\, ref \subseteq h_2 \,.\, ref$
(5) $\forall n. \neg(n \notin h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 \,.\, mark) \vee n \,.\, (h_3 \,.\, ref) = n \,.\, (h_2 \,.\, ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live \,.\, (h_0 \,.\, ref) \not\subseteq live \,.\, (h_3 \,.\, ref)$

WTC:

(9) $\forall h.\ h \,.\, ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

Example

$Types : Heap, Obj$ $\quad$ $Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Consts : h_0, \dots, h_3 : Heap, root, live : Obj$ $\quad$ $Funs : tc_{H.ref} : Heap \to Obj \times Obj$

$F :$

(1) $h_1 . mark = \emptyset$
(2) $h_0 . ref \subseteq h_1 . ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$
(4) $h_1 . ref \subseteq h_2 . ref$
(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

$$Appr_1 : \quad root \underbrace{\dots \xrightarrow{R} \dots}_{\geq 1} live + \text{TC-Ind}$$

Example

$Types : Heap, Obj$

$Consts : h_0, \dots, h_3 : Heap, root, live : Obj$

$Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Funs : tc_{H.ref} : Heap \to Obj \times Obj$

F :

(1) $h_1 . mark = \emptyset$
(2) $h_0 . ref \subseteq h_1 . ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$
(4) $h_1 . ref \subseteq h_2 . ref$
(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

$Appr_1$: $root \underbrace{\dots \xrightarrow{R} \dots}_{\geq 1} live$ + TC-Ind

$Appr_2$: $\forall R, S : Rel.\ R \subseteq S \to R^+ \subseteq S^+$

Example

$Types : Heap, Obj$ $\quad$ $Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq: Heap \times Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$ $\quad$ $Funs : tc_{H.ref} : Heap \to Obj \times Obj$

$F :$

(1) $h_1 . mark = \emptyset$
(2) $h_0 . ref \subseteq h_1 . ref$
(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$
(4) $h_1 . ref \subseteq h_2 . ref$
(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$
(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$
(7) $(root, live) \in tc_{H.ref}(h_0)$
(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$
(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

$Appr_1 :$ $root \underbrace{\ldots \xrightarrow{R} \ldots}_{\geq 1} live$ + TC-Ind

$Appr_2 :$ $\forall R, S : Rel.\ R \subseteq S \to R^+ \subseteq S^+$

$Appr_3 :$ detection and injection of *R-path invariants*

Example

$Types : Heap, Obj$ $\quad$ $Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq : Heap \times Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$ $\quad$ $Funs : tc_{H.ref} : Heap \rightarrow Obj \times Obj$

$F :$

(1) $h_1 . mark = \emptyset$

(2) $h_0 . ref \subseteq h_1 . ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$

(4) $h_1 . ref \subseteq h_2 . ref$

(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC: (9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

The only *essential* R-path p:
$(root, live) \in tc_{H.ref}(h_0)$

$Appr_1 :$ $\quad root \underbrace{\ldots \xrightarrow{R} \ldots}_{\geq 1} live$ + TC-Ind

$Appr_2 :$ $\quad \forall R, S : Rel.\ R \subseteq S \rightarrow R^+ \subseteq S^+$

$Appr_3 :$ detection and injection of *R-path invariants*

Example

Types : *Heap*, *Obj*

Consts : $h_0, \ldots, h_3$: *Heap*, *root*, *live* : *Obj*

Rels : $ref \subseteq Heap \times Obj \times Obj$, $mark \subseteq: Heap \times Obj$

Funs : $tc_{H.ref} : Heap \rightarrow Obj \times Obj$

F :

(1) $h_1 . mark = \emptyset$

(2) $h_0 . ref \subseteq h_1 . ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$

(4) $h_1 . ref \subseteq h_2 . ref$

(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

The only *essential R*-path *p*:
$(root, live) \in tc_{H.ref}(h_0)$

p-invariant $\varphi[n]$:

$Appr_1$: $root \underbrace{\ldots \xrightarrow{R} \ldots}_{\geq 1} live$ + TC-Ind

$Appr_2$: $\forall R, S : Rel.\ R \subseteq S \rightarrow R^+ \subseteq S^+$

$Appr_3$: detection and injection of *R-path invariants*

Example

$Types : Heap, Obj$ $\quad$ $Rels : ref \subseteq Heap \times Obj \times Obj, mark \subseteq : Heap \times Obj$

$Consts : h_0, \ldots, h_3 : Heap, root, live : Obj$ $\quad$ $Funs : tc_{H.ref} : Heap \to Obj \times Obj$

$F :$

(1) $h_1 . mark = \emptyset$

(2) $h_0 . ref \subseteq h_1 . ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$

(4) $h_1 . ref \subseteq h_2 . ref$

(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

The only *essential* R-path p:
$(root, live) \in tc_{H.ref}(h_0)$

p-invariant $\varphi[n]$:
$n \in h_2 . mark$

$Appr_1 :$ $\quad root \underbrace{\ldots \xrightarrow{R} \ldots}_{\geq 1} live$ + TC-Ind

$Appr_2 :$ $\quad \forall R, S : Rel.\ R \subseteq S \to R^+ \subseteq S^+$

$Appr_3 :$ $\quad$ detection and injection of *R-path invariants*

Example

Types : *Heap*, *Obj*

Consts : $h_0, \ldots, h_3$: *Heap*, *root*, *live* : *Obj*

Rels : $ref \subseteq Heap \times Obj \times Obj$, $mark \subseteq: Heap \times Obj$

Funs : $tc_{H.ref} : Heap \to Obj \times Obj$

F :

(1) $h_1 . mark = \emptyset$

(2) $h_0 . ref \subseteq h_1 . ref$

(3) $\forall n. \neg((root, n) \in tc_{H.ref}(h_1)) \vee n \in h_2 . mark$

(4) $h_1 . ref \subseteq h_2 . ref$

(5) $\forall n. \neg(n \notin h_2 . mark) \vee n . (h_3 . ref) = \emptyset$

(6) $\forall n. \neg(n \in h_2 . mark) \vee n . (h_3 . ref) = n . (h_2 . ref)$

(7) $(root, live) \in tc_{H.ref}(h_0)$

(8) $live . (h_0 . ref) \not\subseteq live . (h_3 . ref)$

WTC:

(9) $\forall h.\ h . ref \subseteq tc_{H.ref}(h)$

(10) $\forall h.\ Transitive(tc_{H.ref}(h))$

The only *essential* R-path p:
$(root, live) \in tc_{H.ref}(h_0)$

p-invariant $\varphi[n]$:
$n \in h_2 . mark$

Modified F (*unsat modulo WTC*):

$F \wedge$
$\forall x.\ (root, x) \in tc_{H.ref}(h_0) \to x \in h_2 . mark$

$Appr_1$: $root \underbrace{\ldots \xrightarrow{R} \ldots}_{\geq 1} live$ + TC-Ind

$Appr_2$: $\forall R, S : Rel.\ R \subseteq S \to R^+ \subseteq S^+$

$Appr_3$: detection and injection of *R-path invariants*

p-Invariants

- Definition (R-invariant)

  Let F be a first-order formula and R a binary relation. Then, a formula $\varphi[x]$ is a (forward) R-invariant with respect to x, F and a theory $\mathcal{T}$ if

$$F \models_{\mathcal{T}} \forall x_1, x_2.\ \varphi[x_1/x] \wedge (x_1, x_2) \in R \rightarrow \varphi[x_2/x].$$

- Definition (p-invariant)

  Let F be a first-order formula, R a binary relation and p an R-path of the form $(a, b) \in tc_R$. Then, a (forward) R-invariant formula $\varphi[x]$ is (forward) p-invariant with respect to x, F and a theory $\mathcal{T}$ if a is ground and

$$F \models_{\mathcal{T}} \forall x_2.\ (a, x_2) \in R \rightarrow \varphi[x_2/x].$$

Main Theorem

Theorem (Main theorem)

Let R be a binary relation, F a first-order relational formula and p a difficult R-path of the form $(a, b) \in tc_R$ in a clause C of F. If F is refutable modulo $\mathcal{T}^{TC\text{-}IND}_{tc_R|(a,b)}$ but satisfiable modulo $\mathcal{T}^{WTC}_{tc_R|(a,b)}$, then there exists a p-invariant $\varphi[x]$ w.r.t. x, $F \setminus C$ and $\mathcal{T}^{WTC}_{tc_R|a,b}$, such that

$$F \setminus C \models_{\mathcal{T}^{WTC}_{tc_R|(a,b)}} \neg\varphi[b/x] \text{ and} \tag{1}$$

$$(\forall x_2.\ (a, x_2) \in tc_R \to \varphi[x_2/x]) \land F \text{ is refutable modulo } \mathcal{T}^{WTC}_{tc_R|(a,b)}. \tag{2}$$

Path-Invariant Search Space

$F^{ini} \leftarrow CNF(F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
for $p := (p_s, p_e) \in tc_R \in EP^n$ **do**
  Fix p_s for forward p-inv (p_e for backward)
  if $p_s \in \mathit{Ground}$ **then**
    for $\varphi[x_{1:n}] \subsetneq (F^{ini} \setminus C_p)$ **do**
      for $x_i \in \{x_{1:n}\}$ *with* $p_e \sqsubseteq \mathit{type}(x_i)$ **do**
        (1) Check $\varphi[x_{1:n}]$ of p-invariant w.r.t. p_s, x_i, F^{ret}
        (2) Check abstractions of $\varphi[x_{1:n}]$
  else
    Create ground $p' := (p'_s, p'_e) \in tc_R$ via inst. up to comp. 1
    if $(\forall p'.\ \mathit{unsat}(F[p'/p]|^n_{p'})) \wedge \mathit{sat}(F|^n_p)$ **then**
      Further/General techniques are needed
if $\forall p : EP^n.\ \mathit{unsat}(F|^n_p)$ **then**
  $n \leftarrow n + 1$
return F

Evaluation

Benchmarks	Result	All/Uns/Ess Paths	Che. *p*-inv	Inj. *p*-inv	Time
addrbook-addIdempotent	proved	5 / 2 / 0	0	0	0,08
addrbook-delUndoesAdd	proved	5 / 2 / 0	0	0	0,10
addrbooktrace-addIdempotent	proved	23 / 17 / 0	0	0	0,25
addrbooktrace-delUndoesAdd	proved	20 / 14 / 0	0	0	0,21
addrbooktrace-lookupYields-use	proved	22 / 13 / 0	0	0	0,24
grandpa-noSelfFather	proved	6 / 3 / 0	0	0	0.09
grandpa-noSelfGrandpa	proved	6 / 3 / 0	0	0	0.09
com-theorem1	proved	5 / 2 / 0	0	0	0,18
com-theorem2	proved	5 / 2 / 0	0	0	1.73
com-theorem3	proved	5 / 2 / 0	0	0	0.24
com-theorem4a	proved	5 / 2 / 0	0	0	0.25
com-theorem4b	proved	5 / 2 / 0	0	0	0.13
filesystem-noDirAliases	proved	7 / 4 / 0	0	0	0.12
filesystem-someDir	proved	5 / 3 / 1	2	1	0.15
marksweepgc-soundness1	proved	15 / 9 / 1	38	1	9,29
marksweepgc-soundness2	proved	16 / 10 / 2	75	2	5,92
marksweepgc-completeness	proved	16 / 8 / 2	1021	159	66,58
addrbooktrace-lookupYields-proof	proved	18 / 11 / 2	271	41	79,67
hotelroom-locking	timeout	6 / 3 / 1	–	–	–
javatypes-soundess	timeout	116 / 19 / –	–	–	–

Related Work

- General approaches for proving Alloy
  - Prioni [RMICS'2003] relies on first-order interactive theorem proving
  - Dynamite [TACAS'2007] relies on higher-order theorem proving
  - Kelloy [TACAS'2012] relies on first-order interactive theorem proving
  - In general interactive
  - TC reasoning via general induction and/or general TC lemmas
- Transitive closure specific approaches
  - Functional Reachability [POPL'1983] proposes a *fix* set of FO TC axioms
    – *fix* incompleteness
  - Revisited and extended in [TOPLAS'1998, POPL'2006, Van Eijck'2008]
  - Simulating Reachability [CADE'2005] proposes 3 axiom *schemas* for TC
    – similar to ours in generating context specific TC lemmas
    – no essential paths, no path isolation, not *R*-path directed, only unary formulas, no abstractions
- Well established tools in induction automation
  - e.g. ACL2 and IsaPlanner use similar search algorithms *(lemma calculation)*
    – implementation can profit form their ideas

Conclusion

- A fully automatic approach for proving Alloy spec. involving TC
  - Syntactic detection of unsafe *R*-paths
  - Heuristic detection/isolation of essential *R*-paths – *increases confidence*
  - Complete first-order axiomatization of non essential *R*-paths
  - Directed detection and injection of essential *p*-invariants
- Assumes refutable input formula – *bounded verification helps*
- Rely *completely* on unbounded SMT solving
- Future Works:
  - Reduction of *subsumption* (*p*-invariants, instantiations, abstractions)
  - Prioritization Heuristics (clauses, *p*-invariants, instantiations, abstractions)
  - Further investigation of essential *R*-paths with non ground boundaries
  - More BFS resp. more DFS-breaking criteria

Backup Slides

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
repeat
  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**
    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**
      if $p_g \in Gr$ **then**
        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$
        if $unsat(F)$ **then**
          return F
      else
        $x_{1:n} \leftarrow Var(p_g)$
        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**
          if $sat(F[p'/p]|_{p'}^n)$ **then**
            $p'_g \leftarrow d \; ? \; p'_s : p'_e$
            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$
            if $unsat(F)$ **then**
              return F
        if $(\forall p'. \; unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**
          Further/General techniques are needed
  if $\forall p : EP. \; unsat(F|_p^n)$ **then**
    $n \leftarrow n + 1$
until *F and n are unchanged*;
return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$

repeat

  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|^n_p)\}$ **do**

    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**

      if $p_g \in Gr$ **then**

        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$

        if $unsat(F)$ **then**

          return F

      else

        $x_{1:n} \leftarrow Var(p_g)$

        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**

          if $sat(F[p'/p]|^n_{p'})$ **then**

            $p'_g \leftarrow d \; ? \; p'_s : p'_e$

            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$

            if $unsat(F)$ **then**

              return F

        if $(\forall p'. \; unsat(F[p'/p]|^n_{p'})) \wedge sat(F|^n_p)$ **then**

          Further/General techniques are needed

  if $\forall p : EP. \; unsat(F|^n_p)$ **then**

    $n \leftarrow n + 1$

until *F and n are unchanged*;

return F

Algorithm for Detecting p-invariants

```
F^ini ← CNF(F); F ← F^ini; n ← 1
repeat
    for p := (p_s, p_e) ∈ tc_R ∈ {p ∈ UP(F^ini) | sat(F^ini|^n_p)} do
        for <p_g, d> ∈ {<p_s, 1>, <p_e, −1>} do
            if p_g ∈ Gr then
                F ← pathInv(p, p, p_g, F, F^ini, R, d, n)
                if unsat(F) then
                    return F
            else
                x_{1:n} ← Var(p_g)
                for p' := (p'_s, p'_e) ∈ tc_R ∈ {p[a_{1:n}/x_{1:n}] | a_i ∈ sufGT^1(x_i)}
                do
                    if sat(F[p'/p]|^n_{p'}) then
                        p'_g ← d ? p'_s : p'_e
                        F ← pathInv(p, p', p'_g, F, F^ini, R, d, n)
                        if unsat(F) then
                            return F
                if (∀p'. unsat(F[p'/p]|^n_{p'})) ∧ sat(F|^n_p) then
                    Further/General techniques are needed
    if ∀p : EP. unsat(F|^n_p) then
        n ← n + 1
until F and n are unchanged;
return F
```

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
repeat
    for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**
        for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**
            if $p_g \in Gr$ **then**
                $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$
                if $unsat(F)$ **then**
                    return F
            else
                $x_{1:n} \leftarrow Var(p_g)$
                for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$
                do
                    if $sat(F[p'/p]|_{p'}^n)$ **then**
                        $p'_g \leftarrow d \ ? \ p'_s : p'_e$
                        $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$
                        if $unsat(F)$ **then**
                            return F
                if $(\forall p'.\ unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**
                    Further/General techniques are needed
    if $\forall p : EP.\ unsat(F|_p^n)$ **then**
        $n \leftarrow n + 1$
until *F and n are unchanged*;
return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
repeat
  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|^n_p)\}$ **do**
    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**
      if $p_g \in Gr$ **then**
        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$
        if $unsat(F)$ **then**
          return F
      else
        $x_{1:n} \leftarrow Var(p_g)$
        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**
          if $sat(F[p'/p]|^n_{p'})$ **then**
            $p'_g \leftarrow d \ ? \ p'_s : p'_e$
            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$
            if $unsat(F)$ **then**
              return F
        if $(\forall p'.\ unsat(F[p'/p]|^n_{p'})) \wedge sat(F|^n_p)$ **then**
          Further/General techniques are needed
  if $\forall p : EP.\ unsat(F|^n_p)$ **then**
    $n \leftarrow n + 1$
until *F and n are unchanged*;
return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$

repeat

  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**

    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**

      if $p_g \in Gr$ **then**

        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$

        if $unsat(F)$ **then**

          return F

      else

        $x_{1:n} \leftarrow Var(p_g)$

        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**

          if $sat(F[p'/p]|_{p'}^n)$ **then**

            $p'_g \leftarrow d \ ? \ p'_s : p'_e$

            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$

            if $unsat(F)$ **then**

              return F

        if $(\forall p'. \ unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**

          Further/General techniques are needed

  if $\forall p : EP. \ unsat(F|_p^n)$ **then**

    $n \leftarrow n + 1$

until *F and n are unchanged*;

return F

Algorithm for Detecting *p*-invariants

```
F^ini ← CNF(¬F); F ← F^ini; n ← 1
repeat
    for p := (p_s, p_e) ∈ tc_R ∈ {p ∈ UP(F^ini) | sat(F^ini |^n_p)} do
        for <p_g, d> ∈ {<p_s, 1>, <p_e, −1>} do
            if p_g ∈ Gr then
                F ← pathInv(p, p, p_g, F, F^ini, R, d, n)
                if unsat(F) then
                    return F
            else
                x_{1:n} ← Var(p_g)
                for p' := (p'_s, p'_e) ∈ tc_R ∈ {p[a_{1:n}/x_{1:n}] | a_i ∈ sufGT^1(x_i)}
                do
                    if sat(F[p'/p] |^n_{p'}) then
                        p'_g ← d ? p'_s : p'_e
                        F ← pathInv(p, p', p'_g, F, F^ini, R, d, n)
                        if unsat(F) then
                            return F
                if (∀p'. unsat(F[p'/p] |^n_{p'})) ∧ sat(F |^n_p) then
                    Further/General techniques are needed
    if ∀p : EP. unsat(F |^n_p) then
        n ← n + 1
until F and n are unchanged;
return F
```

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
repeat
  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**
    for $\langle p_g, d\rangle \in \{\langle p_s, 1\rangle, \langle p_e, -1\rangle\}$ **do**
      if $p_g \in Gr$ **then**
        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$
        if $unsat(F)$ **then**
          return F
      else
        $x_{1:n} \leftarrow Var(p_g)$
        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**
          if $sat(F[p'/p]|_{p'}^n)$ **then**
            $p'_g \leftarrow d \; ? \; p'_s : p'_e$
            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$
            if $unsat(F)$ **then**
              return F
        if $(\forall p'.\ unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**
          Further/General techniques are needed
  if $\forall p : EP.\ unsat(F|_p^n)$ **then**
    $n \leftarrow n + 1$
until *F and n are unchanged*;
return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$

repeat

  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**

    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**

      if $p_g \in Gr$ **then**

        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$

        if $unsat(F)$ **then**

          return F

      else

        $x_{1:n} \leftarrow Var(p_g)$

        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**

          if $sat(F[p'/p]|_{p'}^n)$ **then**

            $p'_g \leftarrow d\ ?\ p'_s : p'_e$

            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$

            if $unsat(F)$ **then**

              return F

        if $(\forall p'.\ unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**

          Further/General techniques are needed

  if $\forall p : EP.\ unsat(F|_p^n)$ **then**

    $n \leftarrow n + 1$

until *F and n are unchanged*;

return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$

repeat

  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|_p^n)\}$ **do**

    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**

      if $p_g \in Gr$ **then**

        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$

        if $unsat(F)$ **then**

          return F

      else

        $x_{1:n} \leftarrow Var(p_g)$

        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**

          if $sat(F[p'/p]|_{p'}^n)$ **then**

            $p'_g \leftarrow d\ ?\ p'_s : p'_e$

            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$

            if $unsat(F)$ **then**

              return F

        if $(\forall p'.\ unsat(F[p'/p]|_{p'}^n)) \wedge sat(F|_p^n)$ **then**

          Further/General techniques are needed

  if $\forall p : EP.\ unsat(F|_p^n)$ **then**

    $n \leftarrow n + 1$

until *F and n are unchanged*;

return F

Algorithm for Detecting *p*-invariants

```
F^ini ← CNF(¬F); F ← F^ini; n ← 1
repeat
    for p := (p_s, p_e) ∈ tc_R ∈ {p ∈ UP(F^ini) | sat(F^ini|^n_p)} do
        for <p_g , d> ∈ {<p_s , 1>, <p_e , −1>} do
            if p_g ∈ Gr then
                F ← pathInv(p, p, p_g, F, F^ini, R, d, n)
                if unsat(F) then
                    return F
            else
                x_{1:n} ← Var(p_g)
                for p' := (p'_s, p'_e) ∈ tc_R ∈ {p[a_{1:n}/x_{1:n}] | a_i ∈ sufGT^1(x_i)}
                do
                    if sat(F[p'/p]|^n_{p'}) then
                        p'_g ← d ? p'_s : p'_e
                        F ← pathInv(p, p', p'_g, F, F^ini, R, d, n)
                        if unsat(F) then
                            return F
                if (∀p'. unsat(F[p'/p]|^n_{p'})) ∧ sat(F|^n_p) then
                    Further/General techniques are needed
    if ∀p : EP. unsat(F|^n_p) then
        n ← n + 1
until F and n are unchanged;
return F
```

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$
repeat
  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|^n_p)\}$ **do**
    for $<p_g, d> \in \{<p_s, 1>, <p_e, -1>\}$ **do**
      if $p_g \in Gr$ **then**
        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$
        if $unsat(F)$ **then**
          return F
      else
        $x_{1:n} \leftarrow Var(p_g)$
        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$
        do
          if $sat(F[p'/p]|^n_{p'})$ **then**
            $p'_g \leftarrow d\ ?\ p'_s : p'_e$
            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$
            if $unsat(F)$ **then**
              return F
        if $(\forall p'.\ unsat(F[p'/p]|^n_{p'})) \wedge sat(F|^n_p)$ **then**
          Further/General techniques are needed
  if $\forall p : EP.\ unsat(F|^n_p)$ **then**
    $n \leftarrow n + 1$
until *F and n are unchanged*;
return F

Algorithm for Detecting p-invariants

$F^{ini} \leftarrow CNF(\neg F)$; $F \leftarrow F^{ini}$; $n \leftarrow 1$

repeat

  for $p := (p_s, p_e) \in tc_R \in \{p \in UP(F^{ini}) \mid sat(F^{ini}|^{n}_{p})\}$ **do**

    for $\langle p_g, d\rangle \in \{\langle p_s, 1\rangle, \langle p_e, -1\rangle\}$ **do**

      if $p_g \in Gr$ **then**

        $F \leftarrow pathInv(p, p, p_g, F, F^{ini}, R, d, n)$

        if $unsat(F)$ **then**

          return F

      else

        $x_{1:n} \leftarrow Var(p_g)$

        for $p' := (p'_s, p'_e) \in tc_R \in \{p[a_{1:n}/x_{1:n}] | a_i \in sufGT^1(x_i)\}$ **do**

          if $sat(F[p'/p]|^{n}_{p'})$ **then**

            $p'_g \leftarrow d\ ?\ p'_s : p'_e$

            $F \leftarrow pathInv(p, p', p'_g, F, F^{ini}, R, d, n)$

            if $unsat(F)$ **then**

              return F

        if $(\forall p'.\ unsat(F[p'/p]|^{n}_{p'})) \wedge sat(F|^{n}_{p})$ **then**

          Further/General techniques are needed

  if $\forall p : EP.\ unsat(F|^{n}_{p})$ **then**

    $n \leftarrow n + 1$

until *F and n are unchanged*;

return F

Algorithm for Detecting *p*-invariants

```
F^ini ← CNF(¬F); F ← F^ini; n ← 1
repeat
    for p := (p_s, p_e) ∈ tc_R ∈ {p ∈ UP(F^ini) | sat(F^ini|^n_p)} do
        for <p_g , d> ∈ {<p_s , 1>, <p_e , −1>} do
            if p_g ∈ Gr then
                F ← pathInv(p, p, p_g, F, F^ini, R, d, n)
                if unsat(F) then
                    return F
            else
                x_{1:n} ← Var(p_g)
                for p' := (p'_s, p'_e) ∈ tc_R ∈ {p[a_{1:n}/x_{1:n}] | a_i ∈ sufGT^1(x_i)}
                do
                    if sat(F[p'/p]|^n_{p'}) then
                        p'_g ← d ? p'_s : p'_e
                        F ← pathInv(p, p', p'_g, F, F^ini, R, d, n)
                        if unsat(F) then
                            return F
                if (∀p'. unsat(F[p'/p]|^n_{p'})) ∧ sat(F|^n_p) then
                    Further/General techniques are needed
    if ∀p : EP. unsat(F|^n_p) then
        n ← n + 1
until F and n are unchanged;
return F
```

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**
    for $x_i \in \{x_{1:n}\}$ **do**
        $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**
    $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$
    if $unsat(F[p'/p]|^{n}_{p'})$ **then**
        return F
    for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**
        $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**
  for $x_i \in \{x_{1:n}\}$ **do**
    $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$
    if $unsat(F[p'/p]|^{n}_{p'})$ **then**
      return F
return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**
  $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$
  if $unsat(F[p'/p]|^{n}_{p'})$ **then**
    return F
  for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**
    $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$
    if $unsat(F[p'/p]|^{n}_{p'})$ **then**
      return F
return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : \mathit{Term}, R \subseteq T \times T, d, n : \mathit{Int}$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq \mathit{type}(x_i)$ **do**
    for $x_i \in \{x_{1:n}\}$ **do**
        $F \leftarrow \mathit{concPathInv}(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$
        if $\mathit{unsat}(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : \mathit{Term}, x : \mathit{Var}, R \subseteq T \times T, d, n : \mathit{Int}$

for $\varphi_i[x] \subseteq \varphi$ **do**
    $F \leftarrow \mathit{checkPathInv}(\varphi_i, x, p_g, F, R, d)$
    if $\mathit{unsat}(F[p'/p]|^{n}_{p'})$ **then**
        return F
    for $\varphi'_i[x] \in \mathit{abst}(\varphi_i, F^{ini}, x, R, n)$ **do**
        $F \leftarrow \mathit{checkPathInv}(\varphi'_i, x, p_g, F, R, d)$
        if $\mathit{unsat}(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**

  for $x_i \in \{x_{1:n}\}$ **do**

    $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**

  $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$

  if $unsat(F[p'/p]|^n_{p'})$ **then**

    return F

  for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**

    $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**
    for $x_i \in \{x_{1:n}\}$ **do**
        $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**
    $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$
    if $unsat(F[p'/p]|^{n}_{p'})$ **then**
        return F
    for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**
        $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**

  for $x_i \in \{x_{1:n}\}$ **do**

    $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**

  $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$

  if $unsat(F[p'/p]|^n_{p'})$ **then**

    return F

  for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**

    $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**
    for $x_i \in \{x_{1:n}\}$ **do**
        $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**
    $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$
    if $unsat(F[p'/p]|^{n}_{p'})$ **then**
        return F
    for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**
        $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$
        if $unsat(F[p'/p]|^{n}_{p'})$ **then**
            return F
return F

Algo. 3: *ConcPathInv*

Algorithm for Detecting p-invariants

Data: $p, p', p_g, F, F^{ini} : Term, R \subseteq T \times T, d, n : Int$

for $\varphi[x_{1:n}] \in (F^{ini} \setminus C_p)$ *with* $p_g \sqsubseteq type(x_i)$ **do**

  for $x_i \in \{x_{1:n}\}$ **do**

    $F \leftarrow concPathInv(\varphi, p, p', p_g, F, F^{ini}, x_i, R, d, n)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 2: *pathInv*

Data: $\varphi, p, p', p_g, F, F^{ini} : Term, x : Var, R \subseteq T \times T, d, n : Int$

for $\varphi_i[x] \subseteq \varphi$ **do**

  $F \leftarrow checkPathInv(\varphi_i, x, p_g, F, R, d)$

  if $unsat(F[p'/p]|^n_{p'})$ **then**

    return F

  for $\varphi'_i[x] \in abst(\varphi_i, F^{ini}, x, R, n)$ **do**

    $F \leftarrow checkPathInv(\varphi'_i, x, p_g, F, R, d)$

    if $unsat(F[p'/p]|^n_{p'})$ **then**

      return F

return F

Algo. 3: *ConcPathInv*

Abstraction Rules

Abst$_1$: Variable instantiations with essential ground terms of complexity r (using a modified version of our work [SMT'2013])

Abst$_2$: $\varphi := (l \vee \varphi_{rest}), (\neg l \vee C_{rest}) \in CNF(F) \Longrightarrow$
$\varphi \rightsquigarrow \varphi[C_{rest} \ / \ l]$

Abst$_3$: $\varphi := (\neg\phi(t) \vee \varphi_{rest}), t := p_g \Longrightarrow$
$\varphi \rightsquigarrow \varphi[(\forall x.\ x = t \vee (t, x)^d \in tc_R \rightarrow \phi(x)) \ / \ \phi(t)]$

Abstraction Rules

Abst$_1$: Variable instantiations with essential ground terms of complexity r (using a modified version of our work [SMT'2013])

Abst$_2$: $\varphi := (l \vee \varphi_{rest}), (\neg l \vee C_{rest}) \in CNF(F) \Longrightarrow$
$\varphi \rightsquigarrow \varphi[C_{rest} \ / \ l]$

Abst$_3$: $\varphi := (\neg\phi(t) \vee \varphi_{rest}), t := p_g \Longrightarrow$
$\varphi \rightsquigarrow \varphi[(\forall x.\ x = t \vee (t, x)^d \in tc_R \rightarrow \phi(x)) \ / \ \phi(t)]$

Abstraction Rules

Abst$_1$: Variable instantiations with essential ground terms of complexity r (using a modified version of our work [SMT'2013])

Abst$_2$: $\varphi := (l \vee \varphi_{rest}), (\neg l \vee C_{rest}) \in CNF(F) \Longrightarrow$
$\varphi \rightsquigarrow \varphi[C_{rest} \ / \ l]$

Abst$_3$: $\varphi := (\neg\phi(t) \vee \varphi_{rest}), t := p_g \Longrightarrow$
$\varphi \rightsquigarrow \varphi[(\forall x.\ x = t \vee (t, x)^d \in tc_R \to \phi(x)) \ / \ \phi(t)]$

Abstraction Rules

Abst$_1$: Variable instantiations with essential ground terms of complexity r (using a modified version of our work [SMT'2013])

Abst$_2$: $\varphi := (l \vee \varphi_{rest}), (\neg l \vee C_{rest}) \in CNF(F) \Longrightarrow$
$\varphi \rightsquigarrow \varphi[C_{rest} \ / \ l]$

Abst$_3$: $\varphi := (\neg\phi(t) \vee \varphi_{rest}), t := p_g \Longrightarrow$
$\varphi \rightsquigarrow \varphi[(\forall x.\ x = t \vee (t,x)^d \in tc_R \rightarrow \phi(x)) \ / \ \phi(t)]$