

You Can't Get There From Here:

Reasoning About Reachability in Shape Analysis

Neil Immerman

www.cs.umass.edu/~immerman

Some of this work is due to, or joint with:

Tal Lev-Ami, Bill Hesse, Viktor Kuncak, Tom Reps, Mooly Sagiv, Siddharth Srivastava, and Greta Yorsh

Reachability is Crucial

for reasoning about Linked Data Structures

- **No garbage**, i.e, every node is reachable from a program variable
- **Acyclicity** of data-structure fragments, i.e., from every element reachable from node n , one cannot reach n
- **Effect of procedure calls** when references are passed as arguments: only elements that are reachable from a formal parameter can be modified
- **Termination** of a data-structure traversal: there is a path to a sink-node of the data structure.

One Approach: Using a First-Order Theorem Prover

For each binary relation symbol, f ,
for which we would like to express f^* ,
we add a new binary relation symbol, f_{tc} ,
together with first order axioms about, f_{tc} ,
so that our theorem prover can prove properties about f_{tc} .

- The properties we can prove about f_{tc} will be true of f^* .
- The properties we need concerning f^* , we can **usually** automatically prove about f_{tc} .

Def: A **TC model**, \mathcal{A} , is a model such that if f and f_{tc} are in the vocabulary of \mathcal{A} , then $(f_{tc})^{\mathcal{A}} = (f^{\mathcal{A}})^*$

A set of first-order formulas, Σ , is **TC valid** iff it is true in all TC models.

All the axioms that we consider are TC valid.

Σ is **TC complete** if it is TC valid and for every TC-valid φ , $\Sigma \vdash \varphi$.

Prop: If Σ is TC complete then for all first-order φ ,

$$\Sigma \vdash \varphi \iff \varphi \text{ is TC valid}$$

Some TC-Sound Axioms

$$T_1[f] \equiv \forall u, v (f_{tc}(u, v) \leftrightarrow (u = v) \vee \exists w (f(u, w) \wedge f_{tc}(w, v)))$$

Prop: Any finite and acyclic model of $T_1[f]$ is a TC model.

Proof: Let $\mathcal{A} \models T_1[f]$ where \mathcal{A} is finite and acyclic.

Assume that there is an f -path from a_0 to b .

Since $\mathcal{A} \models T_1[f]$, $\mathcal{A} \models f_{tc}(a_0, b)$.

Conversely, suppose $\mathcal{A} \models f_{tc}(a_0, b)$.

If $a_0 = b$, then there is a path of length 0 from a_0 to b .

Otherwise, exists $a_1 \in |\mathcal{A}|$ s.t. $\mathcal{A} \models f(a_0, a_1) \wedge f_{tc}(a_1, b)$.

Note that $a_1 \neq a_0$ since \mathcal{A} is acyclic. Continuing: a_2, a_3, \dots

Since \mathcal{A} is finite and acyclic, $\mathcal{A} \models f^*(a_0, b)$. □

$$T'_1[f] \equiv \forall u, v (f_{tc}(u, v) \leftarrow (u = v) \vee \exists w (f(u, w) \wedge f_{tc}(w, v)))$$

Prop: If f_{tc} occurs only positively in TC-valid φ
 Then: $T'_1[f] \vdash \varphi$.

Proof: Suppose that $T'_1[f] \not\vdash \varphi$. Let $\mathcal{A} \models T'_1[f] \wedge \neg\varphi$.

Since $\mathcal{A} \models T'_1[f]$, $\mathcal{A} \models f^*(a, b) \Rightarrow \mathcal{A} \models f_{tc}(a, b)$.

Let \mathcal{A}' be \mathcal{A} but with $f_{tc}^{\mathcal{A}'} = (f^{\mathcal{A}})^*$.

\mathcal{A}' is a TC model formed from \mathcal{A} by removing some pairs from $(f_{tc})^{\mathcal{A}}$.

Since $\mathcal{A} \models \neg\varphi$ and f_{tc} occurs only negatively in $\neg\varphi$,
 $\mathcal{A}' \models \neg\varphi$

But φ is TC valid. $\Rightarrow \Leftarrow$



$$T'_1[f] \equiv \forall u, v (f_{tc}(u, v) \leftarrow (u = v) \vee \exists w (f(u, w) \wedge f_{tc}(w, v)))$$

Prop: If f_{tc} occurs only positively in TC-valid φ
Then: $T'_1[f] \vdash \varphi$.

Thus positive TC facts are easy to prove.

The negative TC facts can be more difficult.

An Induction Axiom: **IND**[Z, P, f]

$$\begin{aligned} \forall w(Z(w) \rightarrow P(w)) \quad \wedge \quad \forall u, v(P(u) \wedge f(u, v) \rightarrow P(v)) \\ \rightarrow \quad \forall u, w(Z(w) \wedge f_{tc}(w, u) \rightarrow P(u)) \end{aligned}$$

An Induction Axiom: $\mathbf{IND}[Z, P, f]$

$$\forall w(Z(w) \rightarrow P(w)) \quad \wedge \quad \forall u, v(P(u) \wedge f(u, v) \rightarrow P(v)) \\ \rightarrow \quad \forall u, w(Z(w) \wedge f_{tc}(w, u) \rightarrow P(u))$$

$\mathbf{IND}[Z, P, f]$ is TC Valid

An Induction Axiom: $\mathbf{IND}[Z, P, f]$

$$\forall w(Z(w) \rightarrow P(w)) \quad \wedge \quad \forall u, v(P(u) \wedge f(u, v) \rightarrow P(v)) \\ \rightarrow \quad \forall u, w(Z(w) \wedge f_{\text{tc}}(w, u) \rightarrow P(u))$$

$\mathbf{IND}[Z, P, f]$ is TC Valid

$$T_2[f] \quad \equiv \quad \forall u, v(f_{\text{tc}}(u, v) \leftrightarrow (u = v) \vee \exists w(f_{\text{tc}}(u, w) \wedge f(w, v)))$$

An Induction Axiom: $\mathbf{IND}[Z, P, f]$

$$\forall w(Z(w) \rightarrow P(w)) \quad \wedge \quad \forall u, v(P(u) \wedge f(u, v) \rightarrow P(v)) \\ \rightarrow \quad \forall u, w(Z(w) \wedge f_{\text{tc}}(w, u) \rightarrow P(u))$$

$\mathbf{IND}[Z, P, f]$ is TC Valid

$$T_2[f] \quad \equiv \quad \forall u, v(f_{\text{tc}}(u, v) \leftrightarrow (u = v) \vee \exists w(f_{\text{tc}}(u, w) \wedge f(w, v)))$$

$$T_1[f] \not\vdash T_2[f] \quad \text{and} \quad \mathbf{T}_2[f] \not\vdash T_1[f]$$

But, $\mathbf{IND}, T_1[f] \vdash T_2[f]$ and $\mathbf{IND}, \mathbf{T}_2[f] \vdash T_1[f]$

An Induction Axiom: $\mathbf{IND}[Z, P, f]$

$$\forall w(Z(w) \rightarrow P(w)) \quad \wedge \quad \forall u, v(P(u) \wedge f(u, v) \rightarrow P(v)) \\ \rightarrow \quad \forall u, w(Z(w) \wedge f_{\text{tc}}(w, u) \rightarrow P(u))$$

$\mathbf{IND}[Z, P, f]$ is TC Valid

$$T_2[f] \quad \equiv \quad \forall u, v(f_{\text{tc}}(u, v) \leftrightarrow (u = v) \vee \exists w(f_{\text{tc}}(u, w) \wedge f(w, v)))$$

$$T_1[f] \not\vdash T_2[f] \quad \text{and} \quad \mathbf{T}_2[f] \not\vdash T_1[f]$$

But, $\mathbf{IND}, T_1[f] \vdash T_2[f]$ and $\mathbf{IND}, \mathbf{T}_2[f] \vdash T_1[f]$

$$\mathbf{Trans}[f] \quad \equiv \quad \forall u, v, w(f_{\text{tc}}(u, w) \wedge f_{\text{tc}}(w, v) \rightarrow f_{\text{tc}}(u, v))$$

$$\mathbf{IND}, T_1[f] \vdash \mathbf{Trans}[f]$$

Coloring Axioms

$$\mathbf{NoExit}[A, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f(u, v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f_{tc}(u, v))$$

Coloring Axioms

$$\mathbf{NoExit}[A, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f(u, v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f_{tc}(u, v))$$

$$\mathbf{GoOut}[A, B, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \wedge f(u, v) \rightarrow B(v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \wedge f_{tc}(u, v)) \rightarrow \\ \exists w (B(w) \wedge f_{tc}(u, w) \wedge f_{tc}(w, v))$$

Coloring Axioms

$$\mathbf{NoExit}[A, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f(u, v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f_{tc}(u, v))$$

$$\mathbf{GoOut}[A, B, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \wedge f(u, v) \rightarrow B(v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \wedge f_{tc}(u, v)) \rightarrow \\ \exists w (B(w) \wedge f_{tc}(u, w) \wedge f_{tc}(w, v))$$

$$\mathbf{NewStart}[A, f, f'] \equiv \forall u, v (A(u) \wedge A(v) \wedge f'(u, v) \rightarrow f(u, v)) \rightarrow \\ \forall u, v (f'_{tc}(u, v) \wedge \neg f_{tc}(u, v)) \rightarrow \\ \exists w (\neg A(w) \wedge f'_{tc}(u, w) \wedge f'_{tc}(w, v))$$

Coloring Axioms

$$\mathbf{NoExit}[A, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f(u, v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \rightarrow \neg f_{tc}(u, v))$$

$$\mathbf{GoOut}[A, B, f] \equiv \forall u, v (A(u) \wedge \neg A(v) \wedge f(u, v) \rightarrow B(v)) \rightarrow \\ \forall u, v (A(u) \wedge \neg A(v) \wedge f_{tc}(u, v)) \rightarrow \\ \exists w (B(w) \wedge f_{tc}(u, w) \wedge f_{tc}(w, v))$$

$$\mathbf{NewStart}[A, f, f'] \equiv \forall u, v (A(u) \wedge A(v) \wedge f'(u, v) \rightarrow f(u, v)) \rightarrow \\ \forall u, v (f'_{tc}(u, v) \wedge \neg f_{tc}(u, v)) \rightarrow \\ \exists w (\neg A(w) \wedge f'_{tc}(u, w) \wedge f'_{tc}(w, v))$$

$T_1, \mathbf{IND} \vdash \mathbf{NoExit}, \mathbf{GoOut}, \mathbf{NewStart}$

$T_1, \mathbf{NoExit} \vdash \mathbf{IND}$

Prop: Any finite model of T_1 , **IND** is a TC model.

Prop: Any finite model of T_1 , **IND** is a TC model.

Proof: Let \mathcal{A} be a finite model of T_1 , **IND**.

If there is an f -path from a_0 to b , Then $\mathcal{A} \models f_{tc}(a_0, b)$.

Prop: Any finite model of T_1 , **IND** is a TC model.

Proof: Let \mathcal{A} be a finite model of T_1 , **IND**.

If there is an f -path from a_0 to b , Then $\mathcal{A} \models f_{tc}(a_0, b)$.

Conversely, suppose there is no f path from a to b .

Let $d = \max\{\text{dist}(a, c) \mid f^*(a, c)\}$

Let $C(y) \equiv \text{dist}(a, y) \leq d$

$\mathcal{A} \models \forall u, v (C(u) \wedge \neg C(v) \rightarrow \neg f(u, v))$

Thus, by **NoExit**, $\mathcal{A} \models \forall u, v (C(u) \wedge \neg C(v) \rightarrow \neg f_{tc}(u, v))$

Thus, $\mathcal{A} \models \neg f_{tc}(a, b)$. □

Heuristics for Choosing Instances of Coloring Axioms

coauthor Tal Lev-Ami has programmed some heuristics for generating instances of **NoExit**, **GoOut**, and, **NewStart**, that allow SPASS to automatically prove correctness of a bunch of programs . . .

I'll talk about this at the end if there is time

On TC Completeness

Prop: Let Γ be an r.e. set of TC-valid first-order sentences in a vocabulary with one binary relation, f , and several unary relations. Then Γ is not TC complete.

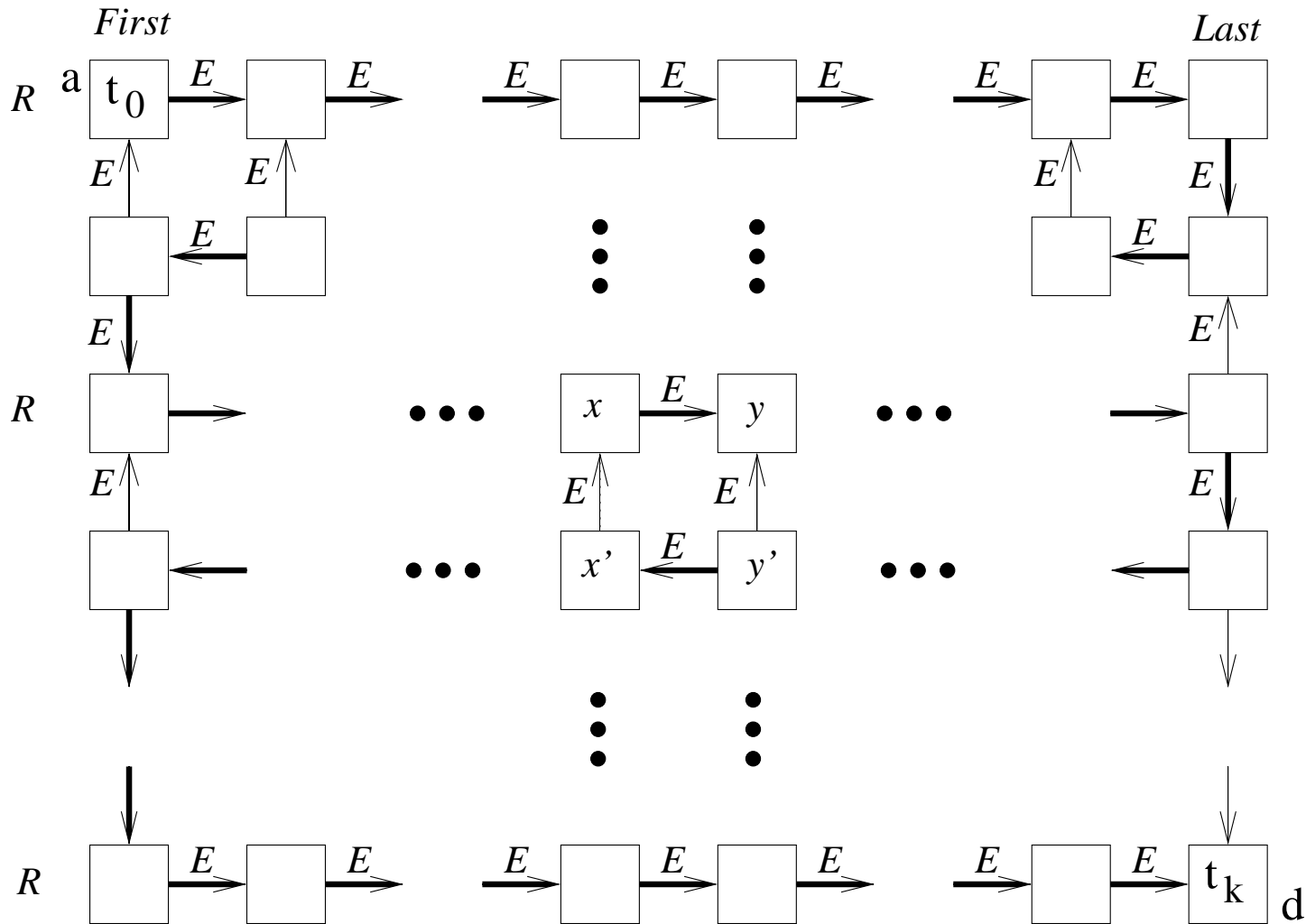
On TC Completeness

Prop: Let Γ be an r.e. set of TC-valid first-order sentences in a vocabulary with one binary relation, f , and several unary relations. Then Γ is not TC complete.

Proof: Using TC we can express the fact that a tiling representation of a Turing machine computation does not reach the halting state. □

On TC Completeness

Prop: Let Γ be an r.e. set of TC-valid first-order sentences in a vocabulary with one binary relation, f , and several unary relations. Then Γ is not TC complete.



Fact: Avron gives an elegant finite axiomatization of the natural numbers using transitive closure, a successor relation and the binary function symbol, “+”. Furthermore, he shows that multiplication is definable in this language. The unique TC model for Avron’s axioms is the standard natural numbers.

Corollary: Let Γ be a set of TC-valid first-order sentences belonging to a fixed level of the arithmetic hierarchy, over a vocabulary including a binary relation symbol and a binary function symbol (or a ternary relation symbol). Then Γ is not TC-complete.

More About TC-Completeness

There is no r.e. set of general TC-complete axioms.

More About TC-Completeness

There is no r.e. set of general TC-complete axioms.

Def: ψ is *TC-valid* wrt Σ iff every TC model of Σ satisfies ψ .

Γ is *TC-complete* wrt Σ iff $\Gamma \cup \Sigma \vdash \psi \Leftrightarrow \psi$ is TC-valid wrt Σ .

More About TC-Completeness

There is no r.e. set of general TC-complete axioms.

Def: ψ is *TC-valid wrt* Σ iff every TC model of Σ satisfies ψ .

Γ is *TC-complete wrt* Σ iff $\Gamma \cup \Sigma \vdash \psi \Leftrightarrow \psi$ is TC-valid wrt Σ .

Wanted: Interesting theories, Σ , for which T_1 , **IND** is TC complete.

When Finiteness is TC Expressible

$$\mathbf{Func}[f] = \forall u, v, w (f(u, v) \wedge f(u, w) \rightarrow v = w)$$

$$\mathbf{Finite} = \mathbf{Func}[s] \wedge \exists x \forall y (s_{tc}(x, y))$$

When Finiteness is TC Expressible

$$\mathbf{Func}[f] = \forall u, v, w (f(u, v) \wedge f(u, w) \rightarrow v = w)$$

$$\mathbf{Finite} = \mathbf{Func}[s] \wedge \exists x \forall y (s_{tc}(x, y))$$

Prop: [Viktor Kuncak] Let Σ be a finite set of formulas, and Γ an r.e., TC-complete axiomatization wrt Σ in a language where finiteness is TC-expressible. Then finite TC-validity for Σ is decidable.

When Finiteness is TC Expressible

$$\mathbf{Func}[f] = \forall u, v, w (f(u, v) \wedge f(u, w) \rightarrow v = w)$$

$$\mathbf{Finite} = \mathbf{Func}[s] \wedge \exists x \forall y (s_{tc}(x, y))$$

Prop: [Viktor Kuncak] Let Σ be a finite set of formulas, and Γ an r.e., TC-complete axiomatization wrt Σ in a language where finiteness is TC-expressible. Then finite TC-validity for Σ is decidable.

Proof: Let ψ be any formula. If ψ is not finite TC-valid wrt Σ , then we can find a finite TC model of Σ where ψ is false.

If ψ is finite TC-valid, then $\Gamma \cup \Sigma \vdash \mathbf{Finite} \rightarrow \psi$,

and we can find this out by systematically generating all proofs from Γ . □

When Finiteness is TC Expressible

$$\mathbf{Func}[f] = \forall u, v, w (f(u, v) \wedge f(u, w) \rightarrow v = w)$$

$$\mathbf{Finite} = \mathbf{Func}[s] \wedge \exists x \forall y (s_{tc}(x, y))$$

Prop: [Viktor Kuncak] Let Σ be a finite set of formulas, and Γ an r.e., TC-complete axiomatization wrt Σ in a language where finiteness is TC-expressible. Then finite TC-validity for Σ is decidable.

Restrict search to cases where finite TC-validity for Σ is decidable.

Cor: There are no r.e. TC-valid axioms for the functional case, even if we restrict to at most two binary relation symbols.

Nelson's Axioms

$$\mathbf{Func}[f] \wedge \forall u, v (f^x(u, v) \leftrightarrow f(u, v) \wedge (u \neq x))$$

$$\mathbf{N1} \quad f_{\text{tc}}^x(u, v) \leftrightarrow (u = v) \vee \exists z ((f^x(u, z) \wedge f_{\text{tc}}^x(z, v)))$$

$$\mathbf{N2} \quad f_{\text{tc}}^x(u, v) \wedge f_{\text{tc}}^x(v, w) \rightarrow f_{\text{tc}}^x(u, w)$$

$$\mathbf{N3} \quad f_{\text{tc}}^x(u, v) \rightarrow f_{\text{tc}}(u, v)$$

$$\mathbf{N4} \quad f_{\text{tc}}^y(u, x) \wedge f_{\text{tc}}^z(u, y) \rightarrow f_{\text{tc}}^z(u, x)$$

$$\mathbf{N5} \quad f_{\text{tc}}(u, x) \rightarrow f_{\text{tc}}^y(u, x) \vee f_{\text{tc}}^x(u, y)$$

$$\mathbf{N6} \quad f_{\text{tc}}^y(u, x) \wedge f_{\text{tc}}^z(u, y) \rightarrow f_{\text{tc}}^z(x, y)$$

$$\mathbf{N7} \quad f(x, u) \wedge f_{\text{tc}}(u, v) \rightarrow f_{\text{tc}}^x(u, v)$$

Nelson's Axioms

$$\mathbf{Func}[f] \wedge \forall u, v (f^x(u, v) \leftrightarrow f(u, v) \wedge (u \neq x))$$

Prop:

- Nelson's Axioms follow from T_1 , **IND**.
- Any finite and functional model of Nelson's axioms is a TC-model.
- Nelson's axioms do not imply **NoExit**.

TC-Completeness for Words

$$v(\Sigma) = \langle 0, \mathit{max}; s^2, s_{\text{tc}}^2, P_\sigma^1 : \sigma \in \Sigma \rangle$$

$$w = \mathit{abbca}$$

$$\mathcal{A}_w = (\{0, \dots, 4\}, 0, 4, P_a = \{0, 4\}, P_b = \{1, 2\}, P_c = \{3\})$$

$$A_{\Sigma w} \equiv A_1 \wedge A_2 \wedge A_3 \wedge A_4$$

$$A_1 \equiv \forall x ((\neg s(x, 0) \wedge \neg s(\mathit{max}, x) \wedge (x \neq 0 \rightarrow \exists y(s(y, x))) \wedge (x \neq \mathit{max} \rightarrow \exists y(s(x, y))))))$$

$$A_2 \equiv \forall xyz (((s(x, y) \wedge s(x, z)) \vee (s(y, x) \wedge s(z, x))) \rightarrow y = z)$$

$$A_3 \equiv \forall x (s_{\text{tc}}(0, x) \wedge s_{\text{tc}}(x, \mathit{max}))$$

$$A_4 \equiv \forall x \left(\bigvee_{\sigma \in \Sigma} (P_\sigma(x) \wedge \bigwedge_{\tau \neq \sigma} \neg P_\tau(x)) \right)$$

Prop: A TC-model of $A_{\Sigma w}$ is exactly a Σ word.

Prop: $\Gamma \cup \{A_{\Sigma w}\} \vdash \forall xy (s(x, y) \rightarrow \neg s_{tc}(y, x))$.

Thm: $\Gamma = \{T_1\} \cup \mathbf{IND}$ is TC complete for words.

Proof Sketch: We will show that Γ proves a version of the McNaughton-Papert Theorem.

Lemma 1: For any $\varphi \in \mathcal{L}(v(\Sigma))$ we can build a DFA

$D_\varphi = (Q_\varphi = \{q_1, \dots, q_n\}, \Sigma, \delta_\varphi, q_1, F_\varphi)$, s.t.

1. Each state, q_i has a formula $q_i(x)$ intuitively meaning $\delta_\varphi^*(q_1, w_0 \dots w_x) = q_i$
2. $\Gamma \cup \mathcal{A}_{\Sigma w}$ proves the following three statements:

$$q_i(0) \quad \leftrightarrow \quad \bigvee_{\sigma \in \Sigma, \delta_\varphi(q_1, \sigma) = q_i} P_\sigma(0)$$

$$\forall u, v \left(s(u, v) \rightarrow \left(q_i(v) \leftrightarrow \bigvee_{\sigma \in \Sigma, \delta_\varphi(q_j, \sigma) = q_i} (P_\sigma(v) \wedge q_j(u)) \right) \right)$$

$$\varphi \leftrightarrow F(max), \quad \text{where } F(u) \equiv \bigvee_{q_i \in F_\varphi} q_i(u)$$

Lemma 2: If $\mathcal{B} \models \Gamma \cup \{A_{\Sigma w}\} \cup \{\varphi\}$. Then, there exists a standard word, w , s.t. $\mathcal{A}_w \models \varphi$.

Lemma 2: If $\mathcal{B} \models \Gamma \cup \{A_{\Sigma w}\} \cup \{\varphi\}$. Then, there exists a standard word, w , s.t. $\mathcal{A}_w \models \varphi$.

Proof: By Lemma 1, have D_φ and $\mathcal{B} \models F_\varphi(max)$.

So \mathcal{B} “believes” there is a path to some $q_f \in F_\varphi$.

Let $C \equiv$ disjunction of all states reachable from q_1 in D_φ .

By **NoExit**, \mathcal{B} knows we must stay in C .

Therefore, $q_f \in C$, i.e., $\mathcal{L}(D_\varphi) \neq \emptyset$

Let $w \in \mathcal{L}(D_\varphi)$. Therefore, $\mathcal{A}_w \models \varphi$. □

Lemma 2: If $\mathcal{B} \models \Gamma \cup \{A_{\Sigma w}\} \cup \{\varphi\}$. Then, there exists a standard word, w , s.t. $\mathcal{A}_w \models \varphi$.

Proof: By Lemma 1, have D_φ and $\mathcal{B} \models F_\varphi(max)$.

So \mathcal{B} “believes” there is a path to some $q_f \in F_\varphi$.

Let $C \equiv$ disjunction of all states reachable from q_1 in D_φ .

By **NoExit**, \mathcal{B} knows we must stay in C .

Therefore, $q_f \in C$, i.e., $\mathcal{L}(D_\varphi) \neq \emptyset$

Let $w \in \mathcal{L}(D_\varphi)$. Therefore, $\mathcal{A}_w \models \varphi$. □

Thus if ψ is TC valid, then $\Gamma \cup \{A_{\Sigma w}\} \vdash \psi$.

It follows that Γ is TC Valid for words.

Conclusions and Directions

- Exploring other heuristics for identifying color classes.
- Exploring the use of additional axiom schemes
- We conjecture that $T_1 + \mathbf{IND}$ is TC-complete for trees.

Reverse Specification

```
Node reverse(Node x){  
    0. Node y = null;  
    1. while (x != null){  
    2.     Node t = x.next;  
    3.     x.next = y;  
    4.     y = x;  
    5.     x = t;  
    6. }  
    7. return y;  
}
```

Precondition: list pointed to by x is acyclic and unshared

Postcondition: list pointed to by return value is acyclic and unshared and the same nodes are reachable from the return value as from the input pointer, but in reverse order.

$$\begin{aligned}
\textit{unique}[z] &\stackrel{\text{def}}{=} \forall v_1, v_2 z(v_1) \wedge z(v_2) \rightarrow v_1 = v_2 \\
\textit{func}[f] &\stackrel{\text{def}}{=} \forall v_1, v_2, v f(v, v_1) \wedge f(v, v_2) \rightarrow v_1 = v_2 \\
\textit{acyclic}[f] &\stackrel{\text{def}}{=} \forall v_1, v_2 \neg f(v_1, v_2) \vee \neg f_{\text{tc}}(v_2, v_1) \\
\textit{unshared}[f] &\stackrel{\text{def}}{=} \forall v_1, v_2, v f(v_1, v) \wedge f(v_2, v) \rightarrow v_1 = v_2 \\
\textit{total}[z_1, z_2, f] &\stackrel{\text{def}}{=} \forall v \exists w (z_1(w) \vee z_2(w)) \wedge f_{\text{tc}}(w, v) \\
r_{x,f}(v) &\stackrel{\text{def}}{=} \exists w (x(w) \wedge f_{\text{tc}}(w, v)) \\
r_{x,\overleftarrow{f}}(v) &\stackrel{\text{def}}{=} \exists w (x(w) \wedge f_{\text{tc}}(v, w)) \\
\textit{pre} &\stackrel{\text{def}}{=} \textit{total}[xe, xe, ne] \wedge \textit{acyclic}[ne] \wedge \textit{unshared}[ne] \wedge \\
&\quad \textit{unique}[xe] \wedge \textit{func}[ne] \\
\textit{post} &\stackrel{\text{def}}{=} \textit{total}[y, y, n] \wedge \textit{acyclic}[n] \wedge \textit{unshared}[n] \wedge \\
&\quad \forall v_1, v_2 (ne(v_1, v_2) \leftrightarrow n(v_2, v_1))
\end{aligned}$$

$$\begin{aligned}
LI[x, y, n] \stackrel{\text{def}}{=} & \text{total}[x, y, n] \wedge \forall v(\neg r_{x,n}(v) \vee \neg r_{y,n}(v)) \wedge \\
& \text{acyclic}[n] \wedge \text{unshared}[n] \\
& \text{unique}[x] \wedge \text{unique}[y] \wedge \text{func}[n] \wedge \\
& \forall v_1, v_2(r_{x,n}(v_1) \rightarrow (ne(v_1, v_2) \leftrightarrow n(v_1, v_2))) \wedge \\
& \forall v_1, v_2(r_{y,n}(v_2) \wedge \neg y(v_1) \rightarrow (ne(v_1, v_2) \leftrightarrow n(v_2, v_1))) \wedge \\
& \forall v_1, v_2, v(y(v_1) \rightarrow (x(v_2) \leftrightarrow ne(v_1, v_2)))
\end{aligned}$$

$$\begin{aligned}
T \stackrel{\text{def}}{=} & \forall v(y'(v) \leftrightarrow x(v)) \wedge \forall v(x'(v) \leftrightarrow \exists w x(w) \wedge n(w, v)) \wedge \\
& \forall v_1, v_2(n'(v_1, v_2) \leftrightarrow \\
& ((n(v_1, v_2) \wedge \neg x(v_1)) \vee (x(v_1) \wedge y(v_2))))
\end{aligned}$$

NoExit $[r_{x',n}, n']$	GoOut $[x, x', n]$	NewStart $[r_{x',n}, n, n']$
NoExit $[r_{x',n'}, n]$	GoOut $[x, y, n']$	NewStart $[r_{x',n'}, n, n']$
NoExit $[r_{y,n}, n']$	NewStart $[r_{x',n}, n', n]$	NewStart $[r_{y,n}, n, n']$
NoExit $[r_{y,n'}, n]$	NewStart $[r_{y,n'}, n, n']$	NewStart $[r_{y,n'}, n', n]$
	NewStart $[r_{x',n'}, n', n]$	NewStart $[r_{y,n}, n', n]$

The instances of coloring axioms used in proving reverse.

Tal's heuristic succeeded in automatically generating the axioms to allow SPASS to prove the postconditions for reverse and the following other examples:

- appending one linked list to another
- mark phase of a mark and sweep garbage collector