Reachability Analysis with Tree Automata

Florian Meßner

January 17, 2018

References


Guillaume Feuillade, Thomas Genet, and Valérie Viet Triem Tong.
Reachability analysis over term rewriting systems.
Journal of Automated Reasoning, 33(3-4):341–383, 2004.

Tree Automata

Tree Automata

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$

Tree Automata

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$
- set of terms (over $\mathcal{F}$ and $\mathcal{X}$) $\mathcal{T}(\mathcal{F}, \mathcal{X})$

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$
- set of terms (over $\mathcal{F}$ and $\mathcal{X}$) $\mathcal{T}(\mathcal{F}, \mathcal{X})$
- set of ground terms (over $\mathcal{F}$) $\mathcal{T}(\mathcal{F})$

Tree Automata

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$
- set of terms (over $\mathcal{F}$ and $\mathcal{X}$) $\mathcal{T}(\mathcal{F}, \mathcal{X})$
- set of ground terms (over $\mathcal{F}$) $\mathcal{T}(\mathcal{F})$
- set of configurations (over $\mathcal{F}$ and $\mathcal{Q}$) $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$

Tree Automata

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$
- set of terms (over $\mathcal{F}$ and $\mathcal{X}$) $\mathcal{T}(\mathcal{F}, \mathcal{X})$
- set of ground terms (over $\mathcal{F}$) $\mathcal{T}(\mathcal{F})$
- set of configurations (over $\mathcal{F}$ and $\mathcal{Q}$) $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$
- $\mathcal{R}^*(E) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in E \text{ s.t. } s \to_{\mathcal{R}}^* t\}$

Tree Automata

Preliminaries

- finite set of function symbols (arity ≥ 0) $\mathcal{F}$
- countable set of variables $\mathcal{X}$
- finite set of state symbols (arity 0) $\mathcal{Q}$ $\quad (\mathcal{Q} \cap \mathcal{F} = \emptyset)$
- set of terms (over $\mathcal{F}$ and $\mathcal{X}$) $\mathcal{T}(\mathcal{F}, \mathcal{X})$
- set of ground terms (over $\mathcal{F}$) $\mathcal{T}(\mathcal{F})$
- set of configurations (over $\mathcal{F}$ and $\mathcal{Q}$) $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$
- $\mathcal{R}^*(E) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in E \text{ s.t. } s \to_{\mathcal{R}}^* t\}$
- set of regular language substitutions $\Sigma(\mathcal{Q}, \mathcal{X})$

Tree Automata

Transition

Tree Automata

Transition

- $c \to q$

Tree Automata

Transition

- $c \to q$
- $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration

Tree Automata

Transition

- $c \to q$
- $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration
- $q \in \mathcal{Q}$ is a state

Tree Automata

Transition

- $c \to q$
- $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration
- $q \in \mathcal{Q}$ is a state

Normalized transition

a transition $c \to q$ where either

Tree Automata

Transition

- $c \to q$
- $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration
- $q \in \mathcal{Q}$ is a state

Normalized transition

a transition $c \to q$ where either

- $c = q' \in \mathcal{Q}$ is a state or

Tree Automata

Transition

- $c \to q$
- $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is a configuration
- $q \in \mathcal{Q}$ is a state

Normalized transition

a transition $c \to q$ where either

- $c = q' \in \mathcal{Q}$ is a state or
- $c = f(q_1, \ldots, q_n)$ where $q_1, \ldots, q_n \in \mathcal{Q}$ are states and $f \in \mathcal{F}, ar(f) = n$

Tree Automata

Example transitions

Tree Automata

Example transitions

- $q_1 \rightarrow q_0$

Tree Automata

Example transitions

- $q_1 \rightarrow q_0$ ✓

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$ ✓

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$ ✓
- $h(q_2, q_3) \to q_1$

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$ ✓
- $h(q_2, q_3) \to q_1$ ✓

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$ ✓
- $h(q_2, q_3) \to q_1$ ✓
- $h(q_1, f(q_2)) \to q_0$

Tree Automata

Example transitions

- $q_1 \to q_0$ ✓
- $f(q_1) \to q_0$ ✓
- $h(q_2, q_3) \to q_1$ ✓
- $h(q_1, f(q_2)) \to q_0$
- $f(g(q_1)) \to q_0$

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$
- set of normalized transitions Δ

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$
- set of normalized transitions Δ
- $\rightarrow_\Delta$ *or* $\rightarrow_\mathcal{A}$: rewriting relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$
- set of normalized transitions Δ
- $\to_\Delta$ *or* $\to_\mathcal{A}$: rewriting relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$
- $\mathcal{L}(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to_\mathcal{A}^* q\}$: language recognized by state q in $\mathcal{A}$

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$
- set of normalized transitions Δ
- $\to_\Delta$ *or* $\to_\mathcal{A}$: rewriting relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$
- $\mathcal{L}(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to_\mathcal{A}^* q\}$: language recognized by state q in $\mathcal{A}$
- $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in \mathcal{Q}_f} \mathcal{L}(\mathcal{A}, q)$ language recognized by automaton

Tree Automata

Bottom-up nondeterministic finite tree automaton

- $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$
- final states $\mathcal{Q}_f \subseteq \mathcal{Q}$
- set of normalized transitions Δ
- $\to_\Delta$ *or* $\to_\mathcal{A}$: rewriting relation on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$
- $\mathcal{L}(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to_\mathcal{A}^* q\}$: language recognized by state q in $\mathcal{A}$
- $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in \mathcal{Q}_f} \mathcal{L}(\mathcal{A}, q)$ language recognized by automaton
- state q is called dead if $\mathcal{L}(\mathcal{A}, q) = \emptyset$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$
- $g(a) \to_\Delta g(q_1) \to_\Delta q_0$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$
- $g(a) \to_\Delta g(q_1) \to_\Delta q_0$
- $f(g(a)) \to_\Delta^* f(q_0) \to_\Delta q_0$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$
- $g(a) \to_\Delta g(q_1) \to_\Delta q_0$
- $f(g(a)) \to_\Delta^* f(q_0) \to_\Delta q_0$
- $\mathcal{L}(\mathcal{A}, q_1) = \{a\}$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$
- $g(a) \to_\Delta g(q_1) \to_\Delta q_0$
- $f(g(a)) \to_\Delta^* f(q_0) \to_\Delta q_0$
- $\mathcal{L}(\mathcal{A}, q_1) = \{a\}$
- $\mathcal{L}(\mathcal{A}, q_0) = \{f(g(a)), f(f(g(a))), \dots\} = \{f^*(g(a))\}$

Tree Automata

Example tree automaton

$\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ where

- $\mathcal{F} = \{f, g, a\}$
- $\mathcal{Q} = \{q_0, q_1, q_2\}$
- $\mathcal{Q}_f = \{q_0\}$
- $\Delta = \{f(q_0) \to q_0, g(q_1) \to q_0, g(q_2) \to q_2, a \to q_1\}$
- $g(a) \to_\Delta g(q_1) \to_\Delta q_0$
- $f(g(a)) \to_\Delta^* f(q_0) \to_\Delta q_0$
- $\mathcal{L}(\mathcal{A}, q_1) = \{a\}$
- $\mathcal{L}(\mathcal{A}, q_0) = \{f(g(a)), f(f(g(a))), \dots\} = \{f^*(g(a))\}$
- $\mathcal{L}(\mathcal{A}, q_2) = \emptyset$ dead state!

Tree Automata Completion

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- successively create $\mathcal{A}_{i+1}$ such that $\mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- successively create $\mathcal{A}_{i+1}$ such that $\mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$
- fixpoint reached when $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}(\mathcal{A}_{k+1})$

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- successively create $\mathcal{A}_{i+1}$ such that $\mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$
- fixpoint reached when $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}(\mathcal{A}_{k+1})$
- completion step finds and joins critical pairs

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- successively create $\mathcal{A}_{i+1}$ such that $\mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$
- fixpoint reached when $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}(\mathcal{A}_{k+1})$
- completion step finds and joins critical pairs
- cp: rule $l \to_{\mathcal{R}} r$, substitution σ and state $q \in \mathcal{Q}$ such that $l\sigma \to^*_{\mathcal{A}_i} q$ and $r\sigma \twoheadrightarrow^*_{\mathcal{A}_i} q$

Completion

- goal: $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- successively create $\mathcal{A}_{i+1}$ such that $\mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$
- fixpoint reached when $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}(\mathcal{A}_{k+1})$
- completion step finds and joins critical pairs
- cp: rule $l \to_{\mathcal{R}} r$, substitution σ and state $q \in \mathcal{Q}$ such that $l\sigma \to^*_{\mathcal{A}_i} q$ and $r\sigma \twoheadrightarrow^*_{\mathcal{A}_i} q$
- normalize then add transition $r\sigma \to q$ to $\mathcal{A}_{i+1}$

Tree Automata

Abstraction function

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- $\alpha : \{f(q_1, \ldots, q_n) | f \in \mathcal{F}, ar(f) = n \text{ and } q_1, \ldots, q_n \in \mathcal{Q}\} \mapsto \mathcal{Q}$

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- $\alpha : \{f(q_1, \ldots, q_n) | f \in \mathcal{F}, ar(f) = n \text{ and } q_1, \ldots, q_n \in \mathcal{Q}\} \mapsto \mathcal{Q}$

Abstraction state

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- $\alpha : \{f(q_1, \ldots, q_n) | f \in \mathcal{F}, ar(f) = n \text{ and } q_1, \ldots, q_n \in \mathcal{Q}\} \mapsto \mathcal{Q}$

Abstraction state

- maps configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- $\alpha : \{f(q_1, \ldots, q_n) | f \in \mathcal{F}, ar(f) = n \text{ and } q_1, \ldots, q_n \in \mathcal{Q}\} \mapsto \mathcal{Q}$

Abstraction state

- maps configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- if $t \in \mathcal{Q}$ then $top_\alpha(t) = t$

Tree Automata

Abstraction function

- maps normalized configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- $\alpha : \{f(q_1, \ldots, q_n) | f \in \mathcal{F}, ar(f) = n \text{ and } q_1, \ldots, q_n \in \mathcal{Q}\} \mapsto \mathcal{Q}$

Abstraction state

- maps configurations in $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ into states
- if $t \in \mathcal{Q}$ then $top_\alpha(t) = t$
- if $t = f(t_1, \ldots, t_n)$ then
  $top_\alpha(t) = \alpha(f(top_\alpha(t_1), \ldots, top_\alpha(t_n)))$

Normalization function

- $Norm_\alpha(s \to q)$ is set of normalized transitions for $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q}), q \in \mathcal{Q}$ and an abstraction function α

Normalization function

- $Norm_\alpha(s \to q)$ is set of normalized transitions for $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q}), q \in \mathcal{Q}$ and an abstraction function α
- if $s = q$ then $Norm_\alpha(s \to q) = \emptyset$

Normalization function

- $Norm_\alpha(s \to q)$ is set of normalized transitions for $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q}), q \in \mathcal{Q}$ and an abstraction function α
- if $s = q$ then $Norm_\alpha(s \to q) = \emptyset$
- if $s \in \mathcal{Q}$ and $s \neq q$ then $Norm_\alpha(s \to q) = \{s \to q\}$

Normalization function

- $Norm_\alpha(s \to q)$ is set of normalized transitions for $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q}), q \in \mathcal{Q}$ and an abstraction function α
- if $s = q$ then $Norm_\alpha(s \to q) = \emptyset$
- if $s \in \mathcal{Q}$ and $s \neq q$ then $Norm_\alpha(s \to q) = \{s \to q\}$
- if $s = f(t_1, \ldots, t_n)$ then $Norm_\alpha(s \to q) =$ $\{f(top_\alpha(t_1), \ldots, top_\alpha(t_n)) \to q\} \cup \bigcup_{i=1}^{n} Norm_\alpha(t_i \to top_\alpha(t_i))$

Tree Automata

One-step completion

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q},\mathcal{X}), l\sigma \to_\Delta^* q} \mathit{Norm}_\alpha(r\sigma \to q)$

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to_\Delta^* q} Norm_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to^*_\Delta q} Norm_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Automaton completion

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to^*_\Delta q} Norm_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Automaton completion

- $\mathcal{A}^0_{\alpha,\mathcal{R}} = \mathcal{A}$

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to_\Delta^* q} Norm_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Automaton completion

- $\mathcal{A}^0_{\alpha,\mathcal{R}} = \mathcal{A}$
- $\mathcal{A}^{n+1}_{\alpha,\mathcal{R}} = \mathcal{C}_{\alpha,\mathcal{R}}(\mathcal{A}^n_{\alpha,\mathcal{R}})$

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to^*_\Delta q} \mathit{Norm}_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Automaton completion

- $\mathcal{A}^0_{\alpha,\mathcal{R}} = \mathcal{A}$
- $\mathcal{A}^{n+1}_{\alpha,\mathcal{R}} = \mathcal{C}_{\alpha,\mathcal{R}}(\mathcal{A}^n_{\alpha,\mathcal{R}})$
- fixpoint: $\mathcal{A}^k_{\alpha,\mathcal{R}} = \mathcal{A}^{k+1}_{\alpha,\mathcal{R}} = \mathcal{A}^*_{\alpha,\mathcal{R}}$ for some $k \in \mathbb{N}$

Tree Automata

One-step completion

- from $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, TRS $\mathcal{R}$ and abstraction function α generate $\mathcal{C}_{\alpha,\mathcal{R}} = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$ where
- $\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, q \in \mathcal{Q}, \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), l\sigma \to^*_\Delta q} \mathit{Norm}_\alpha(r\sigma \to q)$
- $\mathcal{Q}' = \{q | c \to q \in \Delta'\}$

Automaton completion

- $\mathcal{A}^0_{\alpha,\mathcal{R}} = \mathcal{A}$
- $\mathcal{A}^{n+1}_{\alpha,\mathcal{R}} = \mathcal{C}_{\alpha,\mathcal{R}}(\mathcal{A}^n_{\alpha,\mathcal{R}})$
- fixpoint: $\mathcal{A}^k_{\alpha,\mathcal{R}} = \mathcal{A}^{k+1}_{\alpha,\mathcal{R}} = \mathcal{A}^*_{\alpha,\mathcal{R}}$ for some $k \in \mathbb{N}$
- Completed automaton does not exist in general!

Coherence conditions

Left coherence condition

Tree Automaton $\mathcal{A}$ and TRS $\mathcal{R}$ satisfy left-coherence condition if

$\forall \tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}), \forall l \to r \in \mathcal{R}, \forall q \in \mathcal{Q}$:

$l\tau \to_{\Delta}^{*} q \Rightarrow \exists \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ s.t. $l\tau \to_{\Delta}^{*} l\sigma \to_{\Delta}^{*} q$

Coherence conditions

Left coherence condition

Tree Automaton $\mathcal{A}$ and TRS $\mathcal{R}$ satisfy left-coherence condition if

$\forall \tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}), \forall l \to r \in \mathcal{R}, \forall q \in \mathcal{Q}$:

$l\tau \to_{\Delta}^{*} q \Rightarrow \exists \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ s.t. $l\tau \to_{\Delta}^{*} l\sigma \to_{\Delta}^{*} q$

Simple left coherence

Coherence conditions

Left coherence condition

Tree Automaton $\mathcal{A}$ and TRS $\mathcal{R}$ satisfy left-coherence condition if

$\forall \tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}), \forall l \to r \in \mathcal{R}, \forall q \in \mathcal{Q}$:

$l\tau \to_{\Delta}^{*} q \Rightarrow \exists \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ s.t. $l\tau \to_{\Delta}^{*} l\sigma \to_{\Delta}^{*} q$

Simple left coherence

- renaming function $Ren(l) = (l', E)$ where l' is term with renamed variables and E set of equality constraints

Coherence conditions

Left coherence condition

Tree Automaton $\mathcal{A}$ and TRS $\mathcal{R}$ satisfy left-coherence condition if
$\forall \tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}), \forall l \to r \in \mathcal{R}, \forall q \in \mathcal{Q}$:
$l\tau \to_{\Delta}^{*} q \Rightarrow \exists \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ s.t. $l\tau \to_{\Delta}^{*} l\sigma \to_{\Delta}^{*} q$

Simple left coherence

- renaming function $Ren(l) = (l', E)$ where l' is term with renamed variables and E set of equality constraints
- $\forall (x = y) \in E, \forall \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), \forall q, q_x, q_y \in \mathcal{Q}$:
  $l'\sigma \to_{\Delta}^{*} q \wedge \sigma(x) = q_x \neq q_y = \sigma(y) \Longrightarrow$
  $\mathcal{L}(\mathcal{A}, q_x) \cap \mathcal{L}(\mathcal{A}, q_y) = \emptyset$.

Coherence conditions

Left coherence condition

Tree Automaton $\mathcal{A}$ and TRS $\mathcal{R}$ satisfy left-coherence condition if
$\forall \tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}), \forall l \to r \in \mathcal{R}, \forall q \in \mathcal{Q}$:
$l\tau \to_{\Delta}^{*} q \Rightarrow \exists \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ s.t. $l\tau \to_{\Delta}^{*} l\sigma \to_{\Delta}^{*} q$

Simple left coherence

- renaming function $Ren(l) = (l', E)$ where l' is term with renamed variables and E set of equality constraints
- $\forall (x = y) \in E, \forall \sigma \in \Sigma(\mathcal{Q}, \mathcal{X}), \forall q, q_x, q_y \in \mathcal{Q}$:
  $l'\sigma \to_{\Delta}^{*} q \wedge \sigma(x) = q_x \neq q_y = \sigma(y) \Longrightarrow$
  $\mathcal{L}(\mathcal{A}, q_x) \cap \mathcal{L}(\mathcal{A}, q_y) = \emptyset$.
- proven to imply left coherence

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or
- $\forall q \in \mathcal{Q} : \exists t \in \mathcal{T}(\mathcal{F}) : \mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(t)$.

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or
- $\forall q \in \mathcal{Q} : \exists t \in \mathcal{T}(\mathcal{F}) : \mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(t)$.

Coherent abstraction function

Abstraction function α is coherent with $\mathcal{A}$ and $\mathcal{R}$ if

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or
- $\forall q \in \mathcal{Q} : \exists t \in \mathcal{T}(\mathcal{F}) : \mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(t)$.

Coherent abstraction function

Abstraction function α is coherent with $\mathcal{A}$ and $\mathcal{R}$ if

- for all $t \in \mathit{Dom}(\alpha)$ and for all $q \in \mathcal{Q} \cap \mathit{Ran}(\alpha)$

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or
- $\forall q \in \mathcal{Q} : \exists t \in \mathcal{T}(\mathcal{F}) : \mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(t)$.

Coherent abstraction function

Abstraction function α is coherent with $\mathcal{A}$ and $\mathcal{R}$ if

- for all $t \in \mathit{Dom}(\alpha)$ and for all $q \in \mathcal{Q} \cap \mathit{Ran}(\alpha)$
- if $\alpha(t) = q$ then $t \to q \in \mathcal{A}$ and

Coherence conditions

Right coherence

Tree automaton $\mathcal{A}$ and TRS $\mathcal{R}$ are right-coherent if

- $\mathcal{R}$ is right-linear or
- $\forall q \in \mathcal{Q} : \exists t \in \mathcal{T}(\mathcal{F}) : \mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(t)$.

Coherent abstraction function

Abstraction function α is coherent with $\mathcal{A}$ and $\mathcal{R}$ if

- for all $t \in \mathit{Dom}(\alpha)$ and for all $q \in \mathcal{Q} \cap \mathit{Ran}(\alpha)$
- if $\alpha(t) = q$ then $t \to q \in \mathcal{A}$ and
- there is a ground term $t' \in \mathcal{T}(\mathcal{F})$ such that $t' \to_{\mathcal{A}}^* t$ and $\mathcal{L}(\mathcal{A}, q) \subseteq \mathcal{R}^*(\{t'\})$.

Completion Properties

Overapproximation

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then
- completion overapproximates: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then
- completion overapproximates: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Underapproximation

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then
- completion overapproximates: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Underapproximation

- If TRS $\mathcal{R}$ and tree automaton $\mathcal{A}$ are right coherent and

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then
- completion overapproximates: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Underapproximation

- If TRS $\mathcal{R}$ and tree automaton $\mathcal{A}$ are right coherent and
- α is an injective abstraction function coherent with $\mathcal{R}$ and $\mathcal{A}$

Completion Properties

Overapproximation

- If TRS $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ are left coherent then
- completion overapproximates: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Underapproximation

- If TRS $\mathcal{R}$ and tree automaton $\mathcal{A}$ are right coherent and
- α is an injective abstraction function coherent with $\mathcal{R}$ and $\mathcal{A}$

then results of every completion step underapproximate:
$\forall n \in \mathbb{N} : \mathcal{L}(\mathcal{A}^n_{\alpha,\mathcal{R}}) \subseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Completion Properties

Exact case

- If TRS $\mathcal{R}$ and automaton $\mathcal{A}$ satisfy right-coherence and

Completion Properties

Exact case

- If TRS $\mathcal{R}$ and automaton $\mathcal{A}$ satisfy right-coherence and
- completed automaton $\mathcal{A}^*_{\alpha,\mathcal{R}}$ exists and

Completion Properties

Exact case

- If TRS $\mathcal{R}$ and automaton $\mathcal{A}$ satisfy right-coherence and
- completed automaton $\mathcal{A}^*_{\alpha,\mathcal{R}}$ exists and
- $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ satisfy left-coherence and

Completion Properties

Exact case

- If TRS $\mathcal{R}$ and automaton $\mathcal{A}$ satisfy right-coherence and
- completed automaton $\mathcal{A}^*_{\alpha,\mathcal{R}}$ exists and
- $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ satisfy left-coherence and
- abstraction function α is injective and coherent with $\mathcal{R}$ and $\mathcal{A}$

Completion Properties

Exact case

- If TRS $\mathcal{R}$ and automaton $\mathcal{A}$ satisfy right-coherence and
- completed automaton $\mathcal{A}^*_{\alpha,\mathcal{R}}$ exists and
- $\mathcal{R}$ and $\mathcal{A}^*_{\alpha,\mathcal{R}}$ satisfy left-coherence and
- abstraction function α is injective and coherent with $\mathcal{R}$ and $\mathcal{A}$

then completion is exact: $\mathcal{L}(\mathcal{A}^*_{\alpha,\mathcal{R}}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$

Reachability Analysis

Reachability Analysis

(Non-)reachability analysis with tree automata

- compute completed automaton (exact or approximate)

Reachability Analysis

(Non-)reachability analysis with tree automata

- compute completed automaton (exact or approximate)
- intersect with automaton recognizing term t

Reachability Analysis

(Non-)reachability analysis with tree automata

- compute completed automaton (exact or approximate)
- intersect with automaton recognizing term t
- if intersection is empty, t is not reachable

Reachability Analysis

(Non-)reachability analysis with tree automata

- compute completed automaton (exact or approximate)
- intersect with automaton recognizing term t
- if intersection is empty, t is not reachable

Example

Timbuk

Conditional Term Rewriting

Conditional TRS

Conditional TRS

- each rewriting rule $l \to r$ may have conditions

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$

Conditional TRS

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$
- rule $l \to r$ if $c_1 \downarrow c_2$ is enabled for σ if $c_1\sigma \downarrow c_2\sigma$

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$
- rule $l \to r$ if $c_1 \downarrow c_2$ is enabled for σ if $c_1\sigma \downarrow c_2\sigma$
- multiple conditions as conjuction: $c_1 \downarrow c_2 \wedge c_3 \downarrow c_4 \wedge \ldots$

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$
- rule $l \to r$ if $c_1 \downarrow c_2$ is enabled for σ if $c_1\sigma \downarrow c_2\sigma$
- multiple conditions as conjuction: $c_1 \downarrow c_2 \wedge c_3 \downarrow c_4 \wedge \ldots$
- rewriting relation $\to_{\mathcal{R}}^{\downarrow n}$ with

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$
- rule $l \to r$ if $c_1 \downarrow c_2$ is enabled for σ if $c_1\sigma \downarrow c_2\sigma$
- multiple conditions as conjuction: $c_1 \downarrow c_2 \wedge c_3 \downarrow c_4 \wedge \ldots$
- rewriting relation $\to_{\mathcal{R}}^{\downarrow n}$ with
  - $\to_{\mathcal{R}}^{\downarrow 0} = \to_{\mathcal{R}_{nc}}$ relation over non conditional rules

Conditional TRS

- each rewriting rule $l \to r$ may have conditions
- condition $c_1 \downarrow c_2$ requires joinability of terms
- $Var(c_1) \cup Var(c_2) \subseteq Var(l)$
- rule $l \to r$ if $c_1 \downarrow c_2$ is enabled for σ if $c_1\sigma \downarrow c_2\sigma$
- multiple conditions as conjuction: $c_1 \downarrow c_2 \wedge c_3 \downarrow c_4 \wedge \ldots$
- rewriting relation $\to_{\mathcal{R}}^{\downarrow n}$ with
  - $\to_{\mathcal{R}}^{\downarrow 0} = \to_{\mathcal{R}_{nc}}$ relation over non conditional rules
  - $a \to_{\mathcal{R}}^{\downarrow n+1} b \Leftrightarrow a \to_{\mathcal{R}}^{\downarrow 0} b$ or
    $\exists \sigma, p \in Pos(a), (l \to r \text{ if } s \downarrow t) \in \mathcal{R}$ such that
    $a|_p = l\sigma, b = a[r\sigma]_p$ and $\exists u \in \mathcal{T}(\mathcal{F})$ such that
    $s\sigma \to_{\mathcal{R}}^{\downarrow n*} u$ and $t\sigma \to_{\mathcal{R}}^{\downarrow n*} u$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$
- find critical pairs without considering conditions

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$
- find critical pairs without considering conditions
- conditional rule $l \to r$ if $c_1 \downarrow c_2$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$
- find critical pairs without considering conditions
- conditional rule $l \to r$ if $c_1 \downarrow c_2$
  - if $q_{c_1\sigma} \notin \mathcal{Q}_{i,cond}$ or $q_{c_2\sigma} \notin \mathcal{Q}_{i,cond}$ then
    add missing normalized transitions and states to Δ_{i+1} and $\mathcal{Q}_{i+1,cond}$
    $Norm_\alpha(c_1\sigma \to q_{c_1\sigma}) \cup Norm_\alpha(c_2\sigma \to q_{c_2\sigma})$

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$
- find critical pairs without considering conditions
- conditional rule $l \to r$ if $c_1 \downarrow c_2$
  - if $q_{c_1\sigma} \notin \mathcal{Q}_{i,cond}$ or $q_{c_2\sigma} \notin \mathcal{Q}_{i,cond}$ then
    add missing normalized transitions and states to Δ_{i+1} and $\mathcal{Q}_{i+1,cond}$
    $Norm_\alpha(c_1\sigma \to q_{c_1\sigma}) \cup Norm_\alpha(c_2\sigma \to q_{c_2\sigma})$
  - if $\mathcal{L}(\mathcal{A}_i, q_{c_1\sigma}) \cap \mathcal{L}(\mathcal{A}_i, q_{c_2\sigma}) \neq \emptyset$ then condition satisfied
    treat like nonconditional rule

Conditional TRS

Completion algorithm

- input $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$ and CTRS $\mathcal{R}$
- computes $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$
- want fixpoint $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$
- $\mathcal{Q}_i = \mathcal{Q}_0 \cup \mathcal{Q}_{i,new} \cup \mathcal{Q}_{i,cond}$
- find critical pairs without considering conditions
- nonconditional rule $l \to r$
  (or conditional with satisfied condition)
  add normalized transition
  $Norm_\alpha(r\sigma \to q)$

Conditional TRS

Overapproximation of completion

- For $\mathcal{A}_0$ with $\mathcal{L}(\mathcal{A}_0) \supseteq E$ and

Conditional TRS

Overapproximation of completion

- For $\mathcal{A}_0$ with $\mathcal{L}(\mathcal{A}_0) \supseteq E$ and
- left linear CTRS $\mathcal{R}$ and

Conditional TRS

Overapproximation of completion

- For $\mathcal{A}_0$ with $\mathcal{L}(\mathcal{A}_0) \supseteq E$ and
- left linear CTRS $\mathcal{R}$ and
- $\mathcal{A}'$ is result of completion of $\mathcal{A}_0$ with respect to $\mathcal{R}$

Conditional TRS

Overapproximation of completion

- For $\mathcal{A}_0$ with $\mathcal{L}(\mathcal{A}_0) \supseteq E$ and
- left linear CTRS $\mathcal{R}$ and
- $\mathcal{A}'$ is result of completion of $\mathcal{A}_0$ with respect to $\mathcal{R}$

$\mathcal{R}^*(E) \subseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0)) \subseteq \mathcal{L}(\mathcal{A}')$

Conclusion

Conclusion

Conclusion

Conclusion

Conclusion

- completion algorithm

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions
- reachability with tree automata

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions
- reachability with tree automata
- extension to conditional term rewriting

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions
- reachability with tree automata
- extension to conditional term rewriting

Future Work

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions
- reachability with tree automata
- extension to conditional term rewriting

Future Work

- 3-CTRS

Conclusion

Conclusion

- completion algorithm
- completion properties and conditions
- reachability with tree automata
- extension to conditional term rewriting

Future Work

- 3-CTRS
- reachability conditions

The End

Thank you for your attention!