Vérification de systèmes et réécriture de plus en plus efficace

Yohan Boichut
LIFO - Université d'Orléans

JIRC 08, Orléans

9 octobre 2008

Context

Java Bytecode analysis

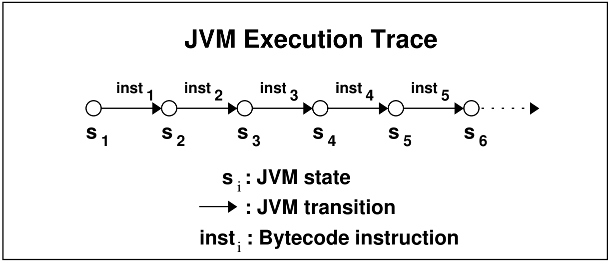
- Rewriting semantics for the Java Bytecode
- Static analysis from reachability analysis in rewriting
  - Tree automata technique [RTA98]
  - Timbuk tool (`http://www.irisa.fr/lande/genet/timbuk`)

Context

Java Bytecode analysis

- Rewriting semantics for the Java Bytecode
- Static analysis from reachability analysis in rewriting
  - Tree automata technique [RTA98]
  - Timbuk tool (`http://www.irisa.fr/lande/genet/timbuk`)

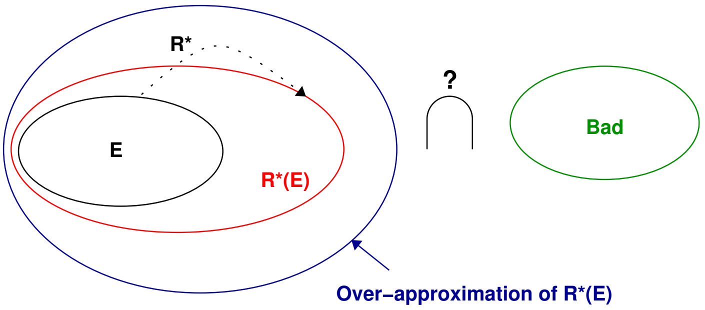
Rewriting Semantics for the Java Bytecode



For a given program `P`

- JVM states as Terms
- Rewrite rules for
  - the Bytecode instructions interpretation (generic rules)
  - the program

Reachability Analysis in Rewriting

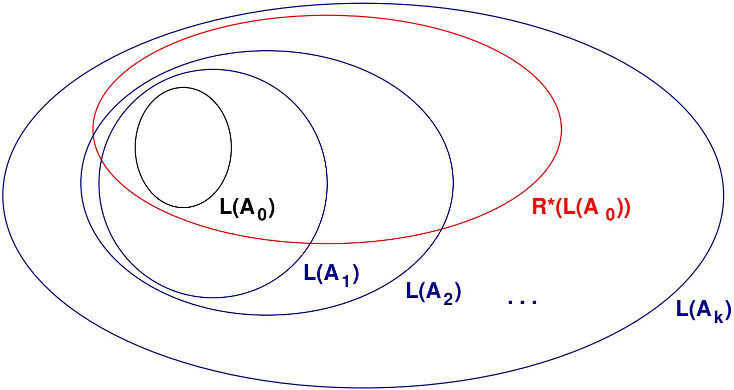


E : initial terms ⟶ JVM initial state

R : term rewriting system ⟶ JVM Transition relation for the given program

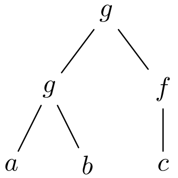
Bad : forbidden terms ⟶ Forbidden JVM states

Tree automata completion

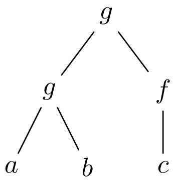


- A set of terms is represented by a tree automaton language
- $\mathcal{A}_{i+1} = \mathcal{A}_i +$ new transitions and states
- Completion stops when a fix point automaton is found

Computing substitutions for a completion step

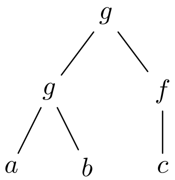


Computing substitutions for a completion step



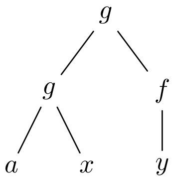
$$\mathcal{A}\quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \end{array}$$

Computing substitutions for a completion step

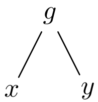


$$\mathcal{A}\quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \end{array}$$

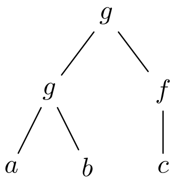
$\mathcal{R}$



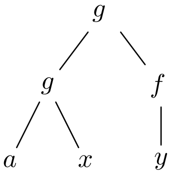
$\rightarrow$



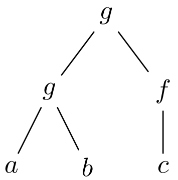
Computing substitutions for a completion step



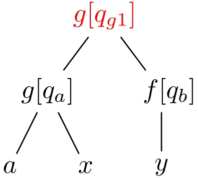
$$
\mathcal{A} \quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



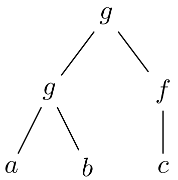
Computing substitutions for a completion step



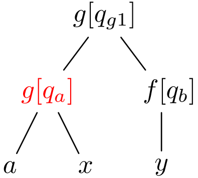
$$
\mathcal{A}\quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
\color{darkred}{g(q_a, q_b)} & \color{darkred}{\rightarrow} & \color{darkred}{q_{g1}} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



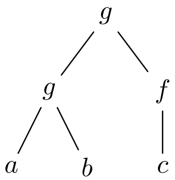
Computing substitutions for a completion step



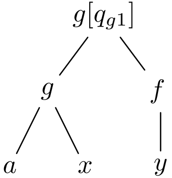
$$\mathcal{A}\quad \begin{array}{lcl} a & \to & q_a \\ b & \to & q_b \\ c & \to & q_c \\ g(q_a, q_b) & \to & q_{g1} \\ f(q_c) & \to & q_f \\ g(q_{g1}, q_f) & \to & q_{g2} \end{array}$$



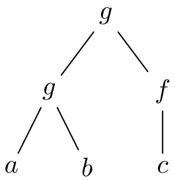
Computing substitutions for a completion step



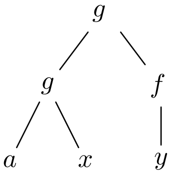
$$\begin{array}{llcl}
 & a & \rightarrow & q_a \\
 & b & \rightarrow & q_b \\
 & c & \rightarrow & q_c \\
\mathcal{A} & g(q_a, q_b) & \rightarrow & q_{g1} \\
 & f(q_c) & \rightarrow & q_f \\
 & g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}$$



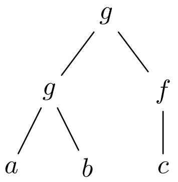
Computing substitutions for a completion step



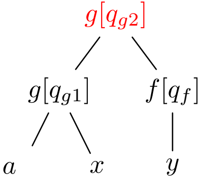
$$
\mathcal{A}\quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



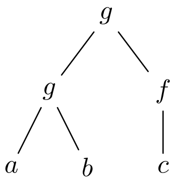
Computing substitutions for a completion step



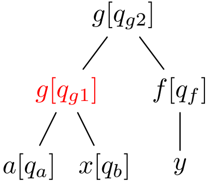
	a	$\rightarrow$	q_a
	b	$\rightarrow$	q_b
	c	$\rightarrow$	q_c
$\mathcal{A}$	$g(q_a, q_b)$	$\rightarrow$	q_{g1}
	$f(q_c)$	$\rightarrow$	q_f
	$g(q_{g1}, q_f)$	$\rightarrow$	q_{g2}



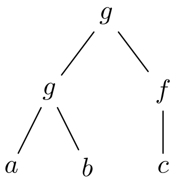
Computing substitutions for a completion step



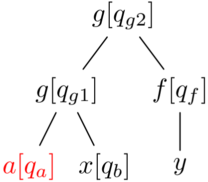
$$
\mathcal{A}\quad
\begin{array}{lcl}
a & \to & q_a \\
b & \to & q_b \\
c & \to & q_c \\
g(q_a, q_b) & \to & q_{g1} \\
f(q_c) & \to & q_f \\
g(q_{g1}, q_f) & \to & q_{g2}
\end{array}
$$



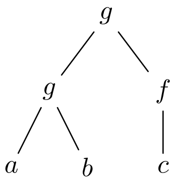
Computing substitutions for a completion step



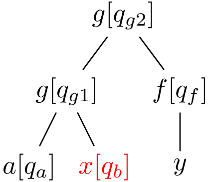
$$\mathcal{A} \quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \end{array}$$



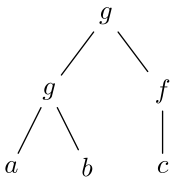
Computing substitutions for a completion step



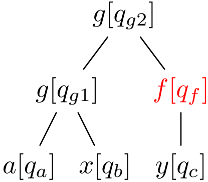
$$
\mathcal{A}\quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



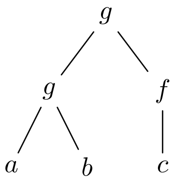
Computing substitutions for a completion step



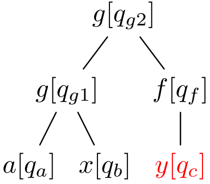
$$\mathcal{A} \quad \begin{array}{lcl} a & \to & q_a \\ b & \to & q_b \\ c & \to & q_c \\ g(q_a, q_b) & \to & q_{g1} \\ f(q_c) & \to & q_f \\ g(q_{g1}, q_f) & \to & q_{g2} \end{array}$$



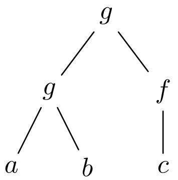
Computing substitutions for a completion step



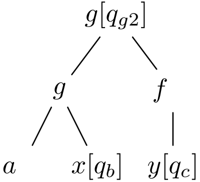
$$
\mathcal{A} \quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



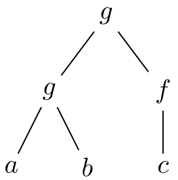
Computing substitutions for a completion step



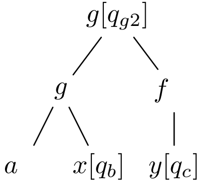
$$\mathcal{A} \quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \end{array}$$



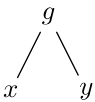
Computing substitutions for a completion step



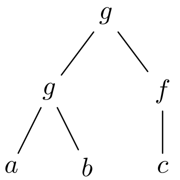
$$\mathcal{A} \quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \end{array}$$



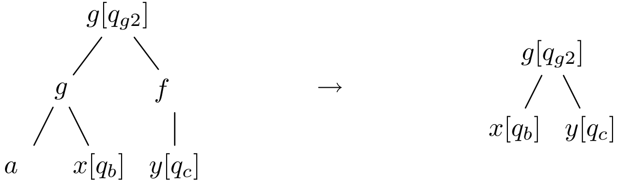
$\rightarrow$



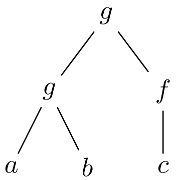
Computing substitutions for a completion step



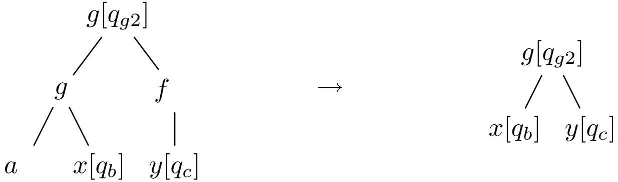
$$
\mathcal{A}\quad
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2}
\end{array}
$$



Computing substitutions for a completion step



$$\mathcal{A}\quad \begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \\ g(q_b, q_c) & \rightarrow & q_{g2} \end{array}$$



Don't forget that it is used for Verification!

- Security protocols: [CADE00,WITS03,CAV05,TFIT06,ICTAC06]
- Java program verification: [RTA 07]

Don't forget that it is used for Verification!

- Security protocols: [CADE00,WITS03,CAV05,TFIT06,ICTAC06]
- Java program verification: [RTA 07]

Don't forget that it is used for Verification!

- Security protocols: [CADE00,WITS03,CAV05,TFIT06,ICTAC06]
- Java program verification: [RTA 07]

Don't forget that it is used for Verification!

- Security protocols: [CADE00,WITS03,CAV05,TFIT06,ICTAC06]
- Java program verification: [RTA 07]

But for the verification of Java programs. . .

- TRS are huge (more than 600 rules for a bubble sort program)
- Computation times with Timbuk may exceed 4 days!

Don't forget that it is used for Verification!

- Security protocols: [CADE00,WITS03,CAV05,TFIT06,ICTAC06]
- Java program verification: [RTA 07]

But for the verification of Java programs. . .

- TRS are huge (more than 600 rules for a bubble sort program)
- Computation times with Timbuk may exceed 4 days!

Goal: Propose practical techniques to solve scalability issues

Reachability Preserving TRS Transformation

Fact: Collecting all possible ground instances of a deep pattern may be expensive

Idea: Transform TRS into simpler TRS

- A simple form for the left hand-side of rules (depth max=2)
  - Flat: $f(x_1, \ldots, x_n)$ or c
  - $f(t_1, \ldots, t_n)$ where each t_i is flat
- Reachability preserving (Terms computed with the original TRS must be also computed by the resulting TRS)

Reachability Preserving TRS Transformation

Fact: Collecting all possible ground instances of a deep pattern may be expensive

Idea: Transform TRS into simpler TRS

- A simple form for the left hand-side of rules (depth max=2)
  - Flat: $f(x_1, \ldots, x_n)$ or c
  - $f(t_1, \ldots, t_n)$ where each t_i is flat
- Reachability preserving (Terms computed with the original TRS must be also computed by the resulting TRS)

Reachability Preserving TRS Transformation

Fact: Collecting all possible ground instances of a deep pattern may be expensive

Idea: Transform TRS into simpler TRS

- A simple form for the left hand-side of rules (depth max=2)
  - Flat: $f(x_1, \ldots, x_n)$ or c
  - $f(t_1, \ldots, t_n)$ where each t_i is flat
- Reachability preserving (Terms computed with the original TRS must be also computed by the resulting TRS)

Reachability Preserving TRS Transformation

Fact: Collecting all possible ground instances of a deep pattern may be expensive

Idea: Transform TRS into simpler TRS

- A simple form for the left hand-side of rules (depth max=2)
  - Flat: $f(x_1, \ldots, x_n)$ or c
  - $f(t_1, \ldots, t_n)$ where each t_i is flat
- Reachability preserving (Terms computed with the original TRS must be also computed by the resulting TRS)

Reachability Preserving TRS Transformation

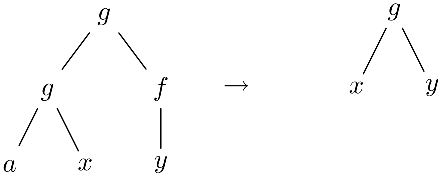
Fact: Collecting all possible ground instances of a deep pattern may be expensive

Idea: Transform TRS into simpler TRS

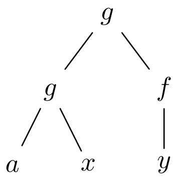
- A simple form for the left hand-side of rules (depth max=2)
  - Flat: $f(x_1, \ldots, x_n)$ or c
  - $f(t_1, \ldots, t_n)$ where each t_i is flat
- Reachability preserving (Terms computed with the original TRS must be also computed by the resulting TRS)

Transformation of the example rule

$\mathcal{R}$

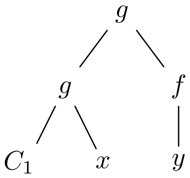


Transformation of the example rule



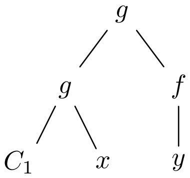
$a \qquad\qquad \rightarrow \quad C_1$

Transformation of the example rule



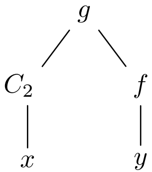
$$a \qquad \rightarrow \qquad C_1$$

Transformation of the example rule



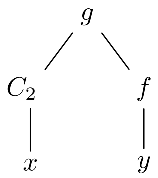
$$\begin{aligned} a &\quad\rightarrow\quad C_1 \\ g(C_1, x) &\quad\rightarrow\quad C_2(x) \end{aligned}$$

Transformation of the example rule



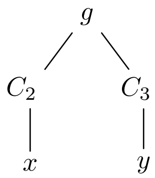
$$\begin{aligned} a &\quad\rightarrow\quad C_1 \\ g(C_1, x) &\quad\rightarrow\quad C_2(x) \end{aligned}$$

Transformation of the example rule



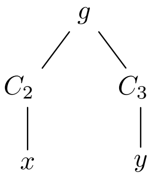
$$\begin{array}{lcl} a & \rightarrow & C_1 \\ g(C_1, x) & \rightarrow & C_2(x) \\ f(y) & \rightarrow & C_3(y) \end{array}$$

Transformation of the example rule



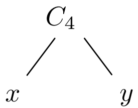
$$\begin{array}{lcl}
a & \to & C_1 \\
g(C_1, x) & \to & C_2(x) \\
f(y) & \to & C_3(y)
\end{array}$$

Transformation of the example rule



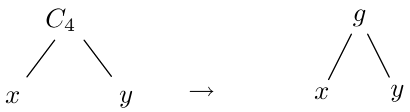
$$
\begin{array}{lcl}
a & \rightarrow & C_1 \\
g(C_1, x) & \rightarrow & C_2(x) \\
f(y) & \rightarrow & C_3(y) \\
g(C_2(x), C_3(y)) & \rightarrow & C_4(x, y)
\end{array}
$$

Transformation of the example rule



$$\begin{array}{lcl}
a & \rightarrow & C_1 \\
g(C_1, x) & \rightarrow & C_2(x) \\
f(y) & \rightarrow & C_3(y) \\
g(C_2(x), C_3(y)) & \rightarrow & C_4(x, y)
\end{array}$$

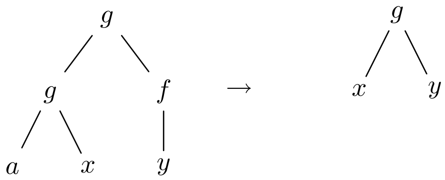
Transformation of the example rule



$$
\begin{array}{lcl}
a & \longrightarrow & C_1 \\
g(C_1, x) & \longrightarrow & C_2(x) \\
f(y) & \longrightarrow & C_3(y) \\
g(C_2(x), C_3(y)) & \longrightarrow & C_4(x, y) \\
C_4(x, y) & \longrightarrow & g(x, y)
\end{array}
$$

Transformation of the example rule

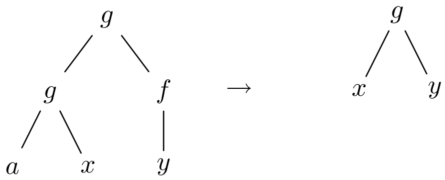
$\mathcal{R}$



$$\phi(\mathcal{R}) \quad \begin{array}{lcl} a & \to & C_1 \\ g(C_1, x) & \to & C_2(x) \\ f(y) & \to & C_3(y) \\ g(C_2(x), C_3(y)) & \to & C_4(x, y) \\ C_4(x, y) & \to & g(x, y) \end{array}$$

Transformation of the example rule

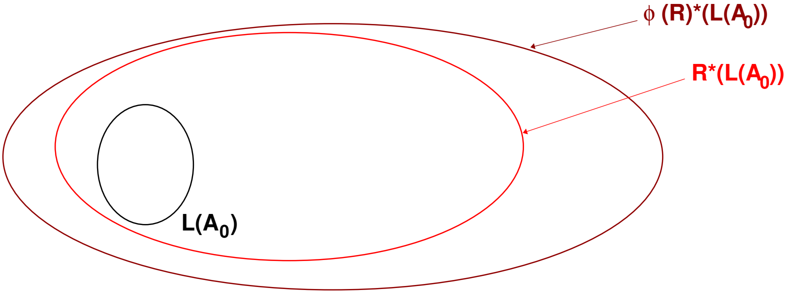
$\mathcal{R}$



$$
\phi(\mathcal{R}) \quad
\begin{array}{lcl}
a & \rightarrow & C_1 \\
g(C_1, x) & \rightarrow & C_2(x) \\
f(y) & \rightarrow & C_3(y) \\
g(C_2(x), C_3(y)) & \rightarrow & C_4(x, y) \\
C_4(x, y) & \rightarrow & g(x, y)
\end{array}
$$

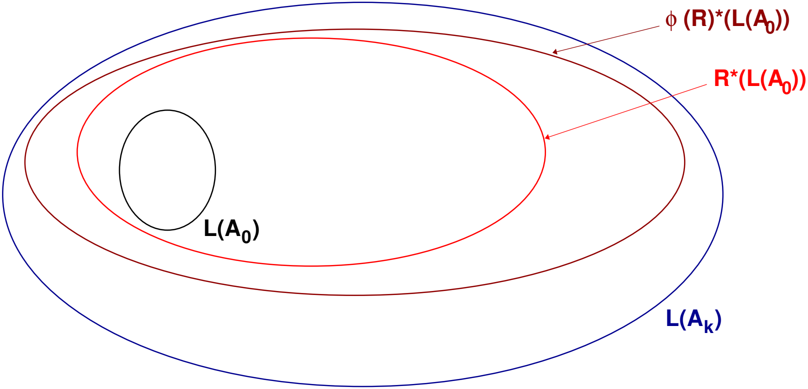
$$\forall t, t' \in \mathcal{T}(\mathcal{F}).t \rightarrow_{\mathcal{R}} t' \implies t \rightarrow^{*}_{\phi(\mathcal{R})} t'$$

Main Result



Main Result

An over-approximation computed for $\phi(\mathcal{R})$ is also an over-approximation for $\mathcal{R}$



Dedicated completion algorithm (1)

Facts

- For each $l \rightarrow r \in \phi(\mathcal{R})$, l does not exceed a depth of 2

  $g(C_2(x), C_3(y)) \rightarrow C_4(x, y)$
- Very close to a direct pattern-matching on transitions

  $g(q_{g1}, q_f) \rightarrow q_{g2}$
- For this transition, the current matching algorithm computes all possible instances from $g(q_{g1}, q_f)$

Can we reduce the substitution computation to a simple pattern-matching problem?

Dedicated completion algorithm (1)

Facts

- For each $l \rightarrow r \in \phi(\mathcal{R})$, l does not exceed a depth of 2
  $$g(C_2(x), C_3(y)) \rightarrow C_4(x, y)$$
- Very close to a direct pattern-matching on transitions
  $$g(q_{g1}, q_f) \rightarrow q_{g2}$$
- For this transition, the current matching algorithm computes all possible instances from $g(q_{g1}, q_f)$

Can we reduce the substitution computation to a simple pattern-matching problem?

Dedicated completion algorithm (1)

Facts

- For each $l \rightarrow r \in \phi(\mathcal{R})$, l does not exceed a depth of 2

  $g(C_2(x), C_3(y)) \rightarrow C_4(x, y)$
- Very close to a direct pattern-matching on transitions

  $g(q_{g1}, q_f) \rightarrow q_{g2}$
- For this transition, the current matching algorithm computes all possible instances from $g(q_{g1}, q_f)$

Can we reduce the substitution computation to a simple pattern-matching problem?

Dedicated completion algorithm (2)

$$\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2} \\
g(q_b, q_c) & \rightarrow & q_{g2} \\
C_1 & \rightarrow & q_a \\
C_3(q_c) & \rightarrow & q_f \\
C_2(q_b) & \rightarrow & q_{g1}
\end{array}$$

$$\begin{array}{lcl}
a & \rightarrow & C_1 \\
g(C_1, x) & \rightarrow & C_2(x) \\
f(y) & \rightarrow & C_3(y) \\
g(C_2(x), C_3(y)) & \rightarrow & C_4(x, y) \\
C_4(x, y) & \rightarrow & g(x, y)
\end{array}$$

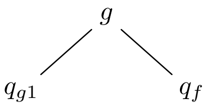
Dedicated completion algorithm (2)

$$\begin{array}{lcl} a & \rightarrow & q_a \\ b & \rightarrow & q_b \\ c & \rightarrow & q_c \\ g(q_a, q_b) & \rightarrow & q_{g1} \\ f(q_c) & \rightarrow & q_f \\ g(q_{g1}, q_f) & \rightarrow & q_{g2} \\ g(q_b, q_c) & \rightarrow & q_{g2} \\ C_1 & \rightarrow & q_a \\ C_3(q_c) & \rightarrow & q_f \\ C_2(q_b) & \rightarrow & q_{g1} \end{array}$$

$$\begin{array}{lcl} a & \rightarrow & C_1 \\ g(C_1, x) & \rightarrow & C_2(x) \\ f(y) & \rightarrow & C_3(y) \\ g(C_2(x), C_3(y)) & \rightarrow & C_4(x, y) \\ C_4(x, y) & \rightarrow & g(x, y) \end{array}$$

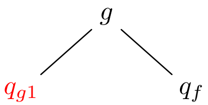
Dedicated completion algorithm (2)

$$
\begin{aligned}
a &\rightarrow q_a \\
b &\rightarrow q_b \\
c &\rightarrow q_c \\
g(q_a, q_b) &\rightarrow q_{g1} \\
f(q_c) &\rightarrow q_f \\
g(q_{g1}, q_f) &\rightarrow q_{g2} \\
g(q_b, q_c) &\rightarrow q_{g2} \\
C_1 &\rightarrow q_a \\
C_3(q_c) &\rightarrow q_f \\
C_2(q_b) &\rightarrow q_{g1}
\end{aligned}
$$



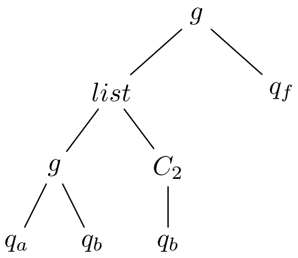
Dedicated completion algorithm (2)

$$
\begin{aligned}
a &\rightarrow q_a \\
b &\rightarrow q_b \\
c &\rightarrow q_c \\
g(q_a, q_b) &\rightarrow q_{g1} \\
f(q_c) &\rightarrow q_f \\
g(q_{g1}, q_f) &\rightarrow q_{g2} \\
g(q_b, q_c) &\rightarrow q_{g2} \\
C_1 &\rightarrow q_a \\
C_3(q_c) &\rightarrow q_f \\
C_2(q_b) &\rightarrow q_{g1}
\end{aligned}
$$



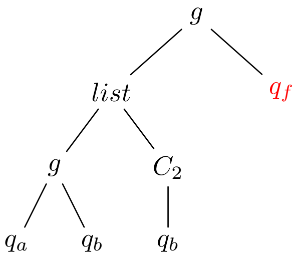
Dedicated completion algorithm (2)

$$
\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2} \\
g(q_b, q_c) & \rightarrow & q_{g2} \\
C_1 & \rightarrow & q_a \\
C_3(q_c) & \rightarrow & q_f \\
C_2(q_b) & \rightarrow & q_{g1}
\end{array}
$$



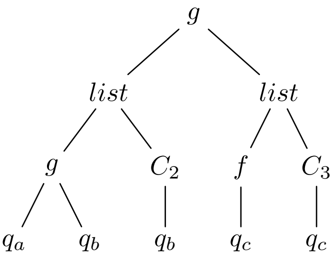
Dedicated completion algorithm (2)

$$\begin{array}{lcl}
a & \rightarrow & q_a \\
b & \rightarrow & q_b \\
c & \rightarrow & q_c \\
g(q_a, q_b) & \rightarrow & q_{g1} \\
f(q_c) & \rightarrow & q_f \\
g(q_{g1}, q_f) & \rightarrow & q_{g2} \\
g(q_b, q_c) & \rightarrow & q_{g2} \\
C_1 & \rightarrow & q_a \\
C_3(q_c) & \rightarrow & q_f \\
C_2(q_b) & \rightarrow & q_{g1}
\end{array}$$



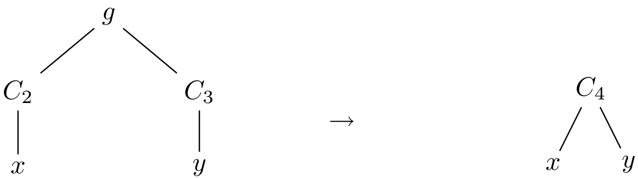
Dedicated completion algorithm (2)

$$
\begin{aligned}
a &\rightarrow q_a \\
b &\rightarrow q_b \\
c &\rightarrow q_c \\
g(q_a, q_b) &\rightarrow q_{g1} \\
f(q_c) &\rightarrow q_f \\
g(q_{g1}, q_f) &\rightarrow q_{g2} \\
g(q_b, q_c) &\rightarrow q_{g2} \\
C_1 &\rightarrow q_a \\
C_3(q_c) &\rightarrow q_f \\
C_2(q_b) &\rightarrow q_{g1}
\end{aligned}
$$



Dedicated completion algorithm (3)

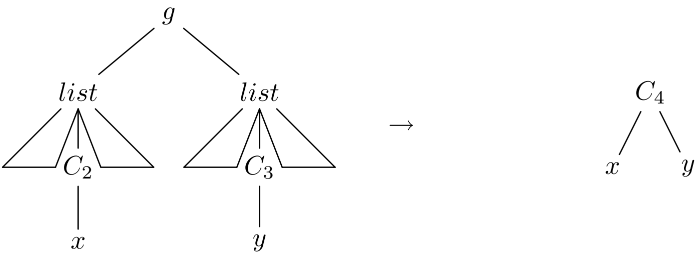
We want to do a completion step with the rule
$g(C_2(x), C_3(y)) \to C_4(x, y)$.



The particular form of the rules allows us to replace the substitution calculus by associative pattern-matching

Dedicated completion algorithm (3)

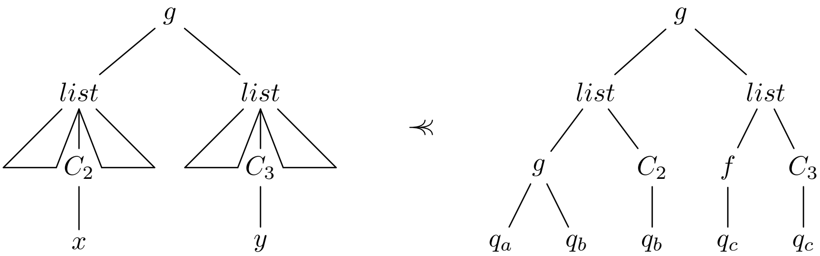
We want to do a completion step with the rule
$g(C_2(x), C_3(y)) \rightarrow C_4(x, y)$.



The particular form of the rules allows us to replace the substitution calculus by associative pattern-matching

Dedicated completion algorithm (3)

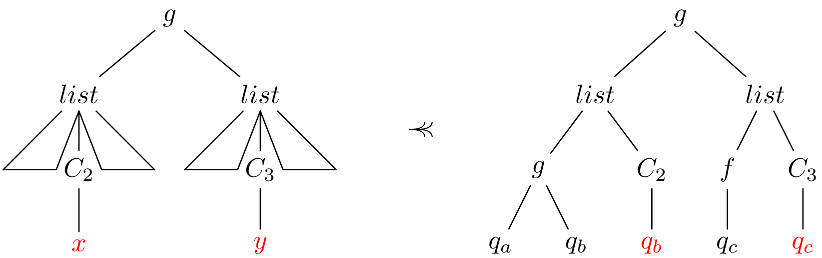
We want to do a completion step with the rule
$g(C_2(x), C_3(y)) \to C_4(x, y)$.



The particular form of the rules allows us to replace the substitution calculus by associative pattern-matching

Dedicated completion algorithm (3)

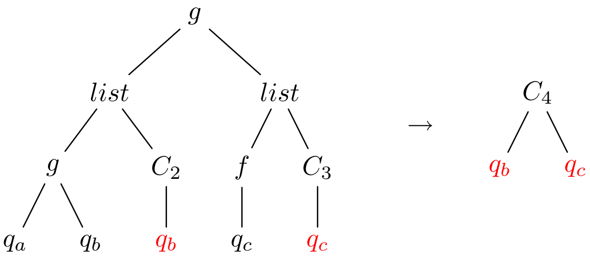
We want to do a completion step with the rule $g(C_2(x), C_3(y)) \to C_4(x, y)$.



The particular form of the rules allows us to replace the substitution calculus by associative pattern-matching

Dedicated completion algorithm (3)

We want to do a completion step with the rule
$g(C_2(x), C_3(y)) \to C_4(x, y)$.



The particular form of the rules allows us to replace the substitution calculus by associative pattern-matching

General schema of the implementation

The Tom language [RTA'07]: Piggybacking Rewriting on top of Java

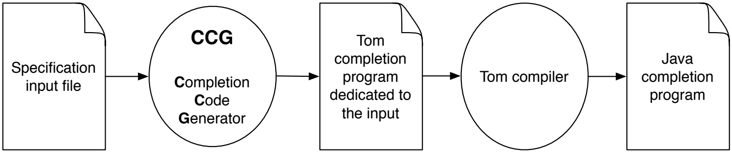
- Efficient support for algebraic terms (hash-consing),
- Pattern-matching (AU theory, variadic operators),
- Expressive strategy language (*à la* ELAN, Stratego).

General schema of the implementation

The Tom language [RTA'07]: Piggybacking Rewriting on top of Java

- Efficient support for algebraic terms (hash-consing),
- Pattern-matching (AU theory, variadic operators),
- Expressive strategy language (*à la* ELAN, Stratego).

Generator of dedicated completion programs written in Tom



Experimental results

	NSPK protocol	View-Only protocol	Java program (chained lists)
TRS size (nb of rules)	13	15	303
Timbuk: Time (secs)	**19.7**	**6420**	**37387**
Tom: Time (secs)	**5.9**	**150**	**303**
Timbuk/Tom	**3**	**40**	**120**

In practice, conclusive analyses with Timbuk are also conclusive with Tom

Experimental results

	NSPK protocol	View-Only protocol	Java program (chained lists)
TRS size (nb of rules)	13	15	303
Timbuk: Time (secs)	**19.7**	**6420**	**37387**
Tom: Time (secs)	**5.9**	**150**	**303**
Timbuk/Tom	**3**	**40**	**120**

In practice, conclusive analyses with Timbuk are also conclusive with Tom

Conclusion

Main results:

- Definition of a reachability preserving transformation on TRS
- Computations of over-approximations using associative pattern-matching
- Implementation in Tom/Java
- A factor 10 in general, and up to 100 on Java examples

Future work:

- Verification of MIDlets
- A better control of approximations
- Using threads to parallelize the completion procedure

Conclusion

Main results:

- Definition of a reachability preserving transformation on TRS
- Computations of over-approximations using associative pattern-matching
- Implementation in Tom/Java
- A factor 10 in general, and up to 100 on Java examples

Future work:

- Verification of MIDlets
- A better control of approximations
- Using threads to parallelize the completion procedure