Automata-Based Abstraction Refinement for μHORS Model Checking

Naoki Kobayashi, Xin Li
The University of Tokyo

This Talk

Efficient model checking algorithm for μHORS (Recursively-typed Higher-Order Recursion Scheme) [Kobayashi, Igarashi, ESOP13]

which has been applied to automated verification of

- functional OO (objected-oriented) and
- multi-threaded higher-order programs

Naoki Kobayashi, Xin Li. "Automata-based abstraction refinement for μHORS model checking." LICS15

Motivation: μHORS Model Checking

[Kobayashi, Igarashi, ESOP13]

- μHORS: a model that is Turing-complete
  (≈ recursively-typed call-by-name λ-calculus)
- Sound procedure based on iterations of type inference, checking and refinement
- Relatively-complete w.r.t. certain condition of types

μHORS +
Tree automata
(for safety property)

→

μHORS
Model Checker
RTRecs

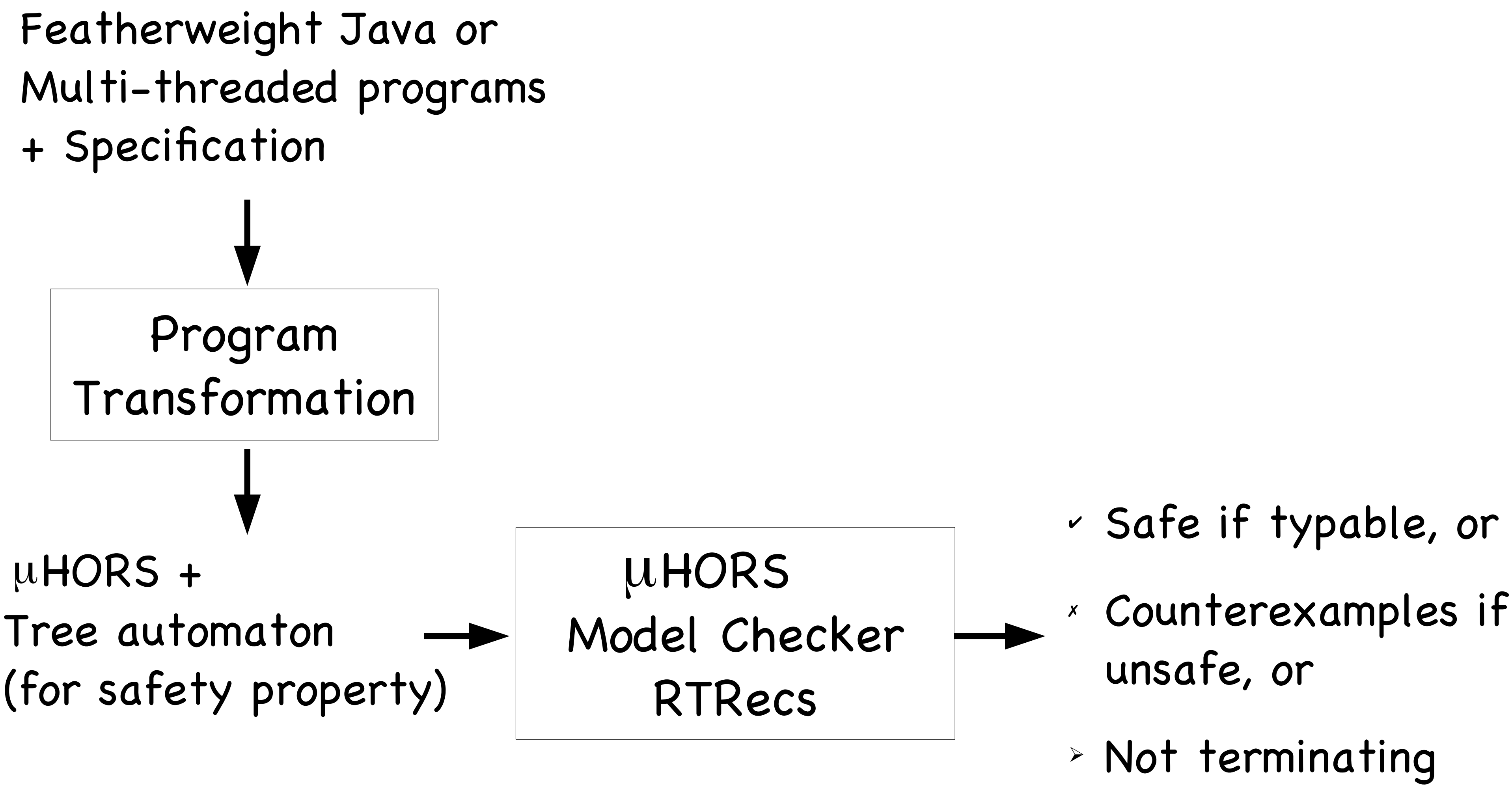
→

- ✓ Safe if typable, or
- ✗ Counterexamples if unsafe, or
- ➢ Not terminating

Motivation: μHORS Model Checking

[Kobayashi, Igarashi, ESOP13]

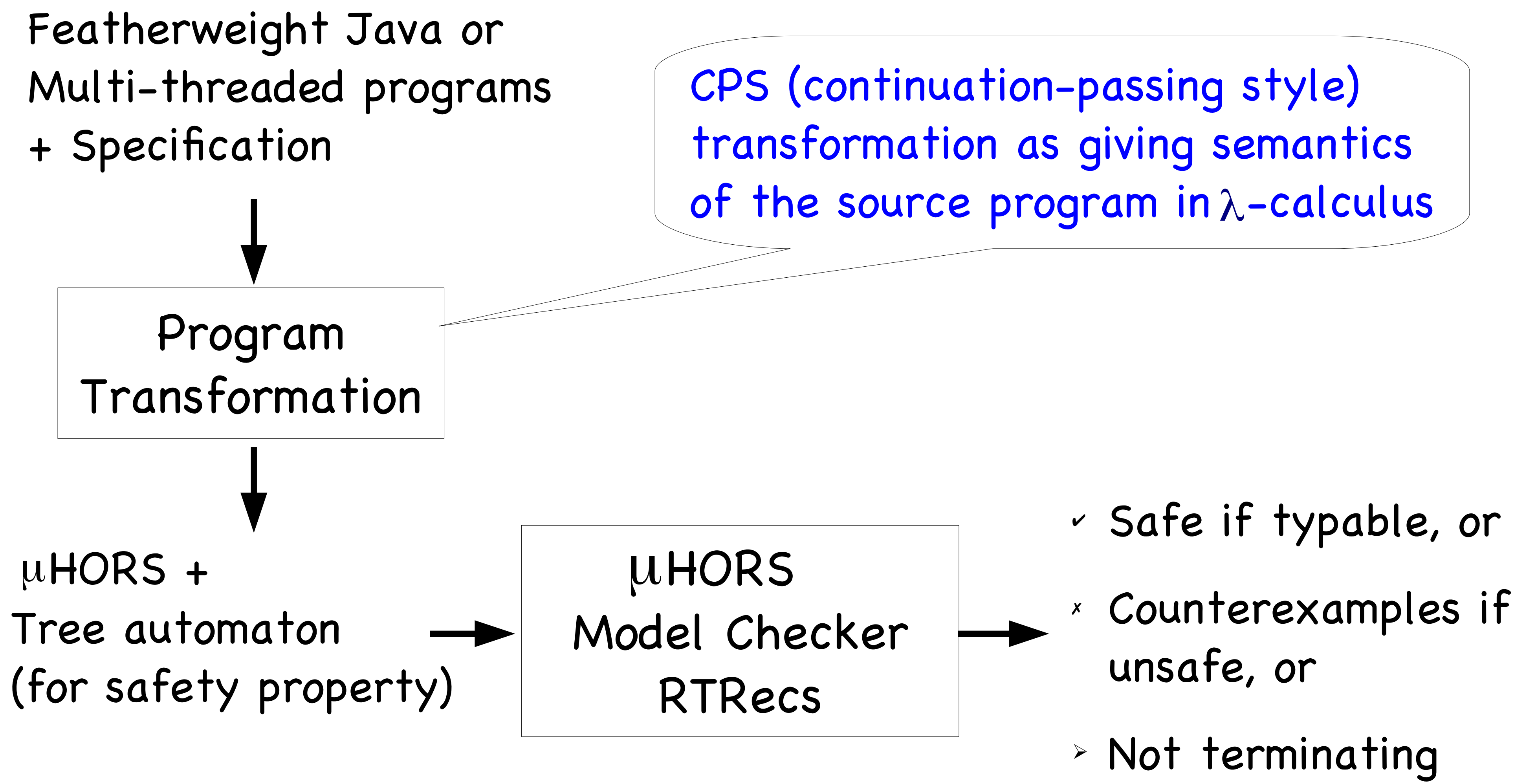
- Applications to OO and multi-threaded programs



Motivation: μHORS Model Checking

[Kobayashi, Igarashi, ESOP13]

- Applications to OO and multi-threaded programs



Limitations of RTRecs

- Does not scale well as the size of the tree automaton states increases
- Counterexample finding by an exhaustive search of the state space is ineffective

Our Contributions

- A sound procedure for μHORS that often scales better by experiments
- Relatively complete w.r.t. certain conditions of (a regular set of) term trees
- Evaluation by verification of OO and multi-threaded boolean programs with recursion

Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Illustrate abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and conclusion

HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \rightarrow F\, c$$
$$F\, x \rightarrow a\, x\, (F\, (b\, x))$$
$$S : o\,,\ \ F : o \rightarrow o$$

HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \to F\, c$$
$$F\, x \to a\, x\,(F\,(b\, x))$$
$$S : o,\ \ F : o \to o$$

tree constructors

Simply-typed

HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \to F\,c$$
$$F\,x \to a\,x\,(F\,(b\,x))$$
$$S : o,\;\; F : o \to o$$

tree constructors

Simply-typed

$$S \longrightarrow F\,c$$

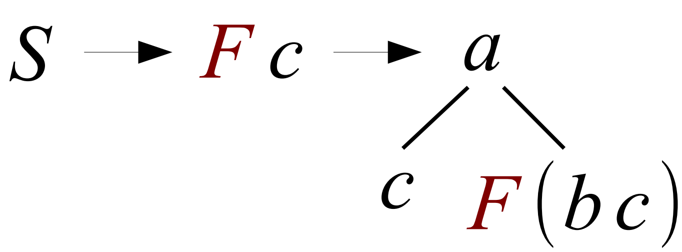
HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \to F\,c$$
$$F\,x \to a\,x\,(F\,(b\,x))$$
$$S : o,\quad F : o \to o$$

tree constructors

Simply-typed



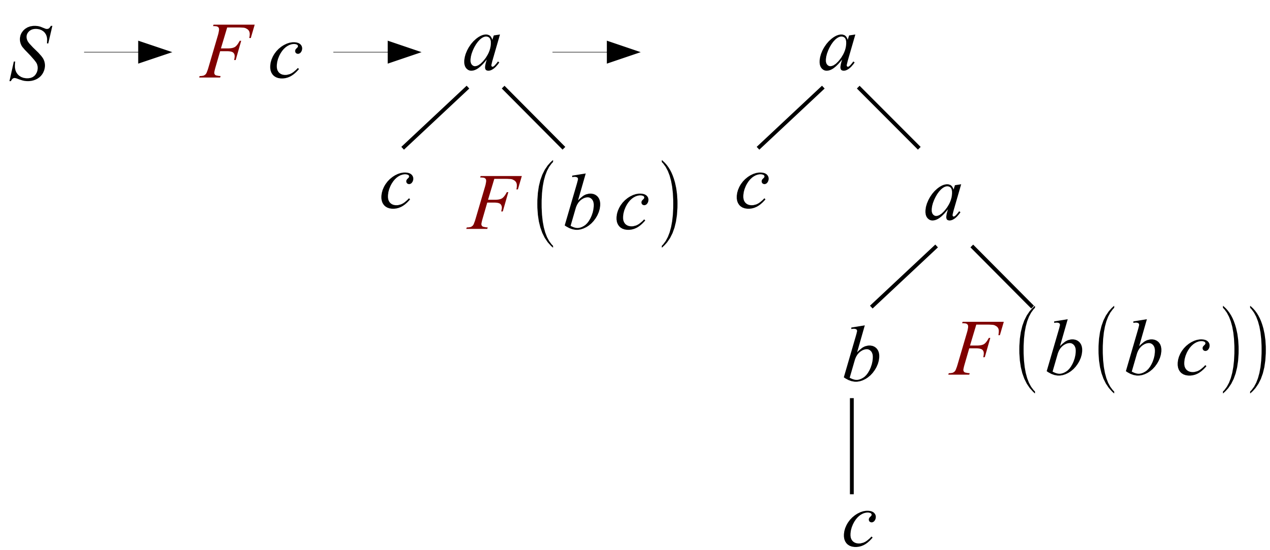
HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \rightarrow F\,c$$
$$F\,x \rightarrow a\,x\,(F\,(b\,x))$$
$$S : o,\quad F : o \rightarrow o$$

tree constructors

Simply-typed



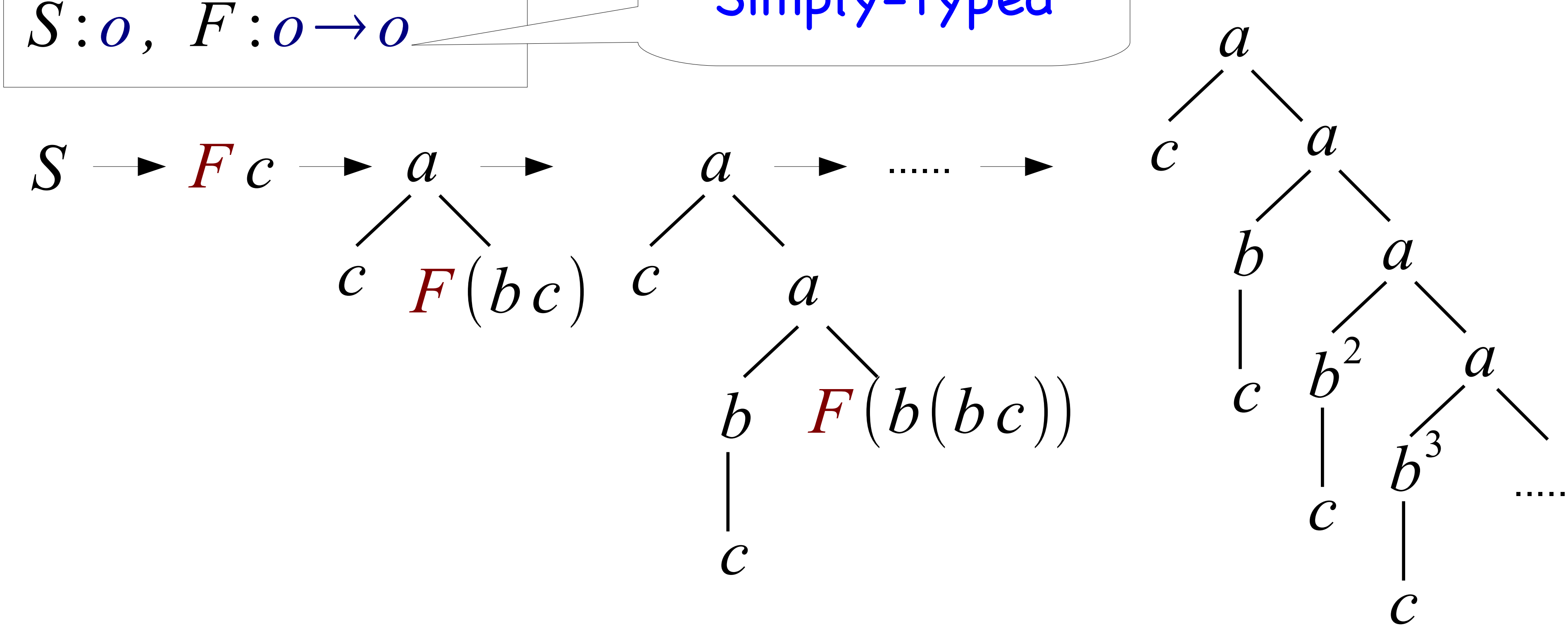
HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \to F\,c$$
$$F\,x \to a\,x\,(F\,(b\,x))$$
$$S : o, \quad F : o \to o$$

tree constructors

Simply-typed



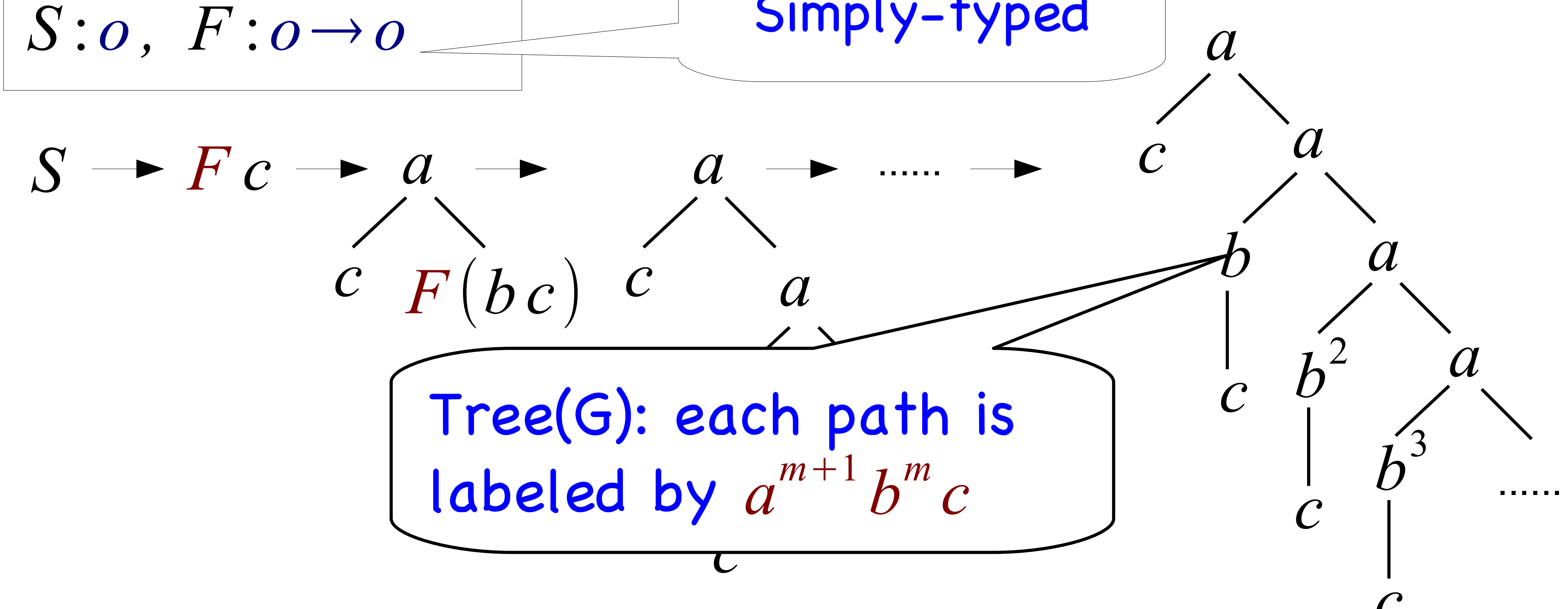
HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees

$$S \to F\, c$$
$$F\, x \to a\, x\, (F\, (b\, x))$$
$$S : o,\quad F : o \to o$$

tree constructors

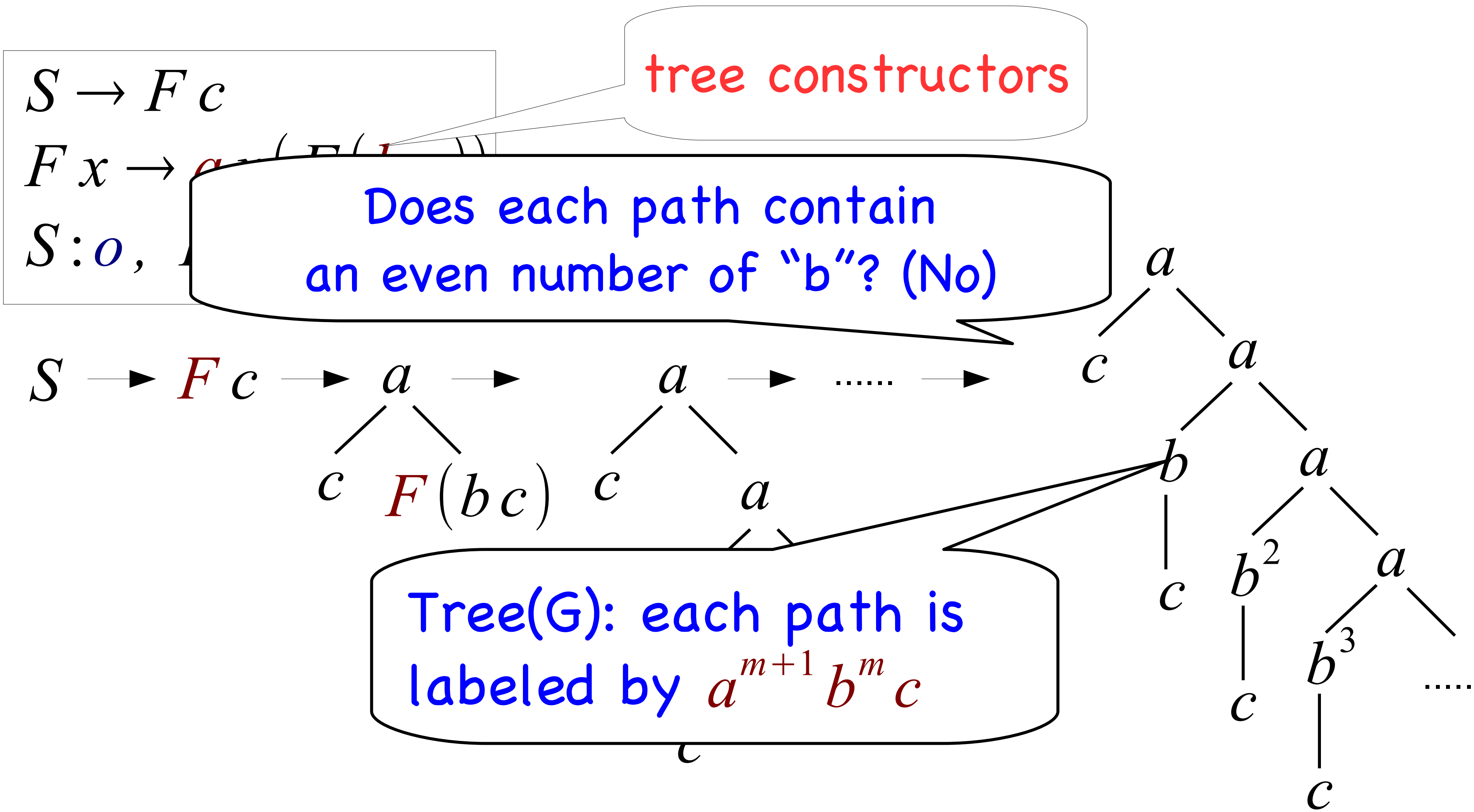
Simply-typed



Tree(G): each path is labeled by $a^{m+1} b^m c$

HORS: Higher-Order Recursion Scheme

Grammar for generating infinite trees



μHORS = HORS + Recursive Types

[Kobayashi, Igarashi, ESOP13]

Recursive types

$$\tau \quad ::= \quad \alpha \mid \tau_1 \to \cdots \to \tau_n \to o \mid \mu\alpha.\tau$$

$$S \to F\,F\,b$$
$$F\,f\,g \to a(g(g\,c))(f\,f\,(B\,g))$$
$$B\,h\,x \to b(h\,x)$$
$$S : o,\, B : (o \to o) \to o \to o$$
$$F : \mu\alpha.\alpha \to (o \to o) \to o$$

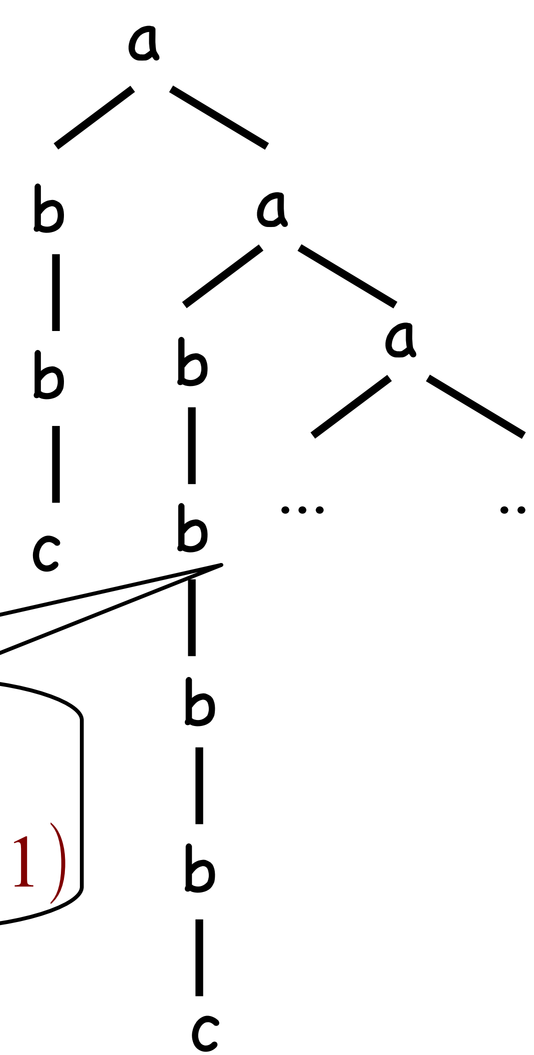
μHORS = HORS + Recursive Types

[Kobayashi, Igarashi, ESOP13]

Recursive types

$$\tau \quad ::= \quad \alpha \mid \tau_1 \to \cdots \to \tau_n \to o \mid \mu\alpha.\tau$$

$$S \to F\,F\,b$$
$$F\,f\,g \to a(g(g\,c))(f\,f\,(B\,g))$$
$$B\,h\,x \to b(h\,x)$$
$$S : o,\ B : (o \to o) \to o \to o$$
$$F : \mu\alpha.\alpha \to (o \to$$



Tree(G): each path is labeled by $a^m b^{2m} c\ (m \geq 1)$

μHORS = HORS + Recursive Types

[Kobayashi, Igarashi, ESOP13]

Recursive types

$$\tau \quad ::= \quad \alpha \mid \tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow o \mid \mu\alpha.\tau$$

$S \rightarrow F\,F\,b$

$F\,f\,g \rightarrow a(g(g\,c))(f\,f\,(B\,g))$

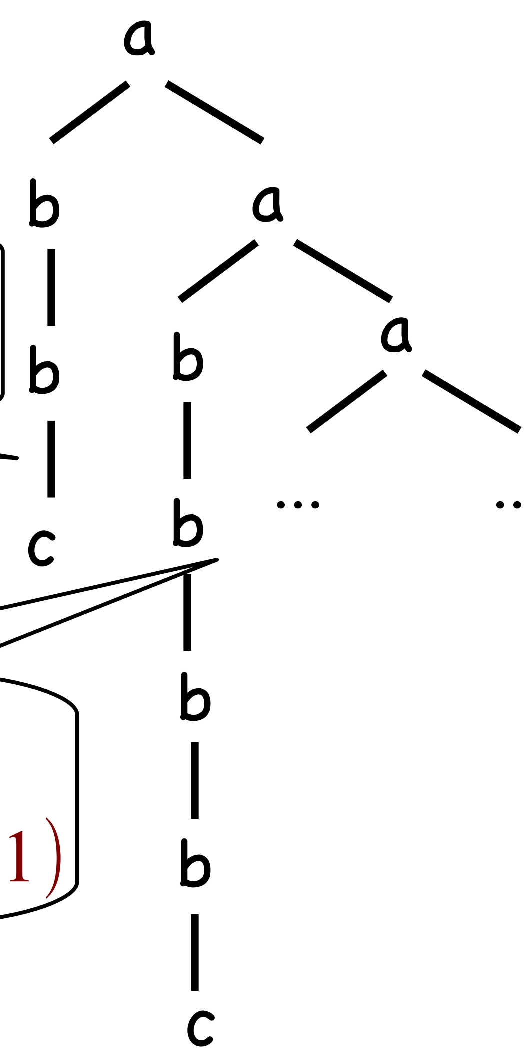
$B\,h\,x \rightarrow b(h\,x)$

$S : o,\ B : (o \rightarrow o) \rightarrow o \rightarrow o$

$F : \mu\alpha.\alpha \rightarrow (o \rightarrow$

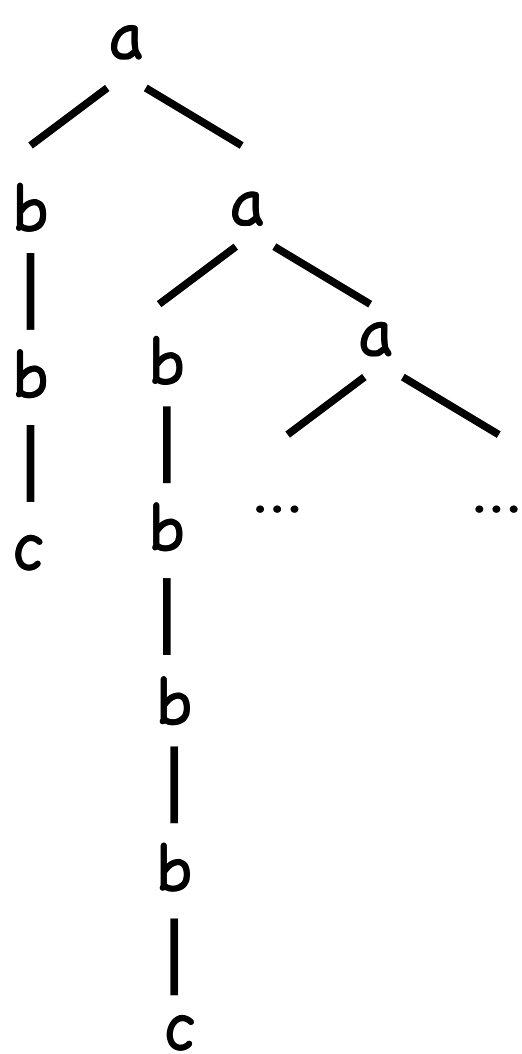
Does each path contain an even number of “b”? (yes)

Tree(G): each path is labeled by $a^m b^{2m} c\ (m \geq 1)$



Trivial Tree Automata [Klaus A., LMCS07]

(top-down deterministic, all states are final)



$A:$

$\delta(q_0, a) = q_0 q_0$

$\delta(q_1, a) = q_1 q_1$

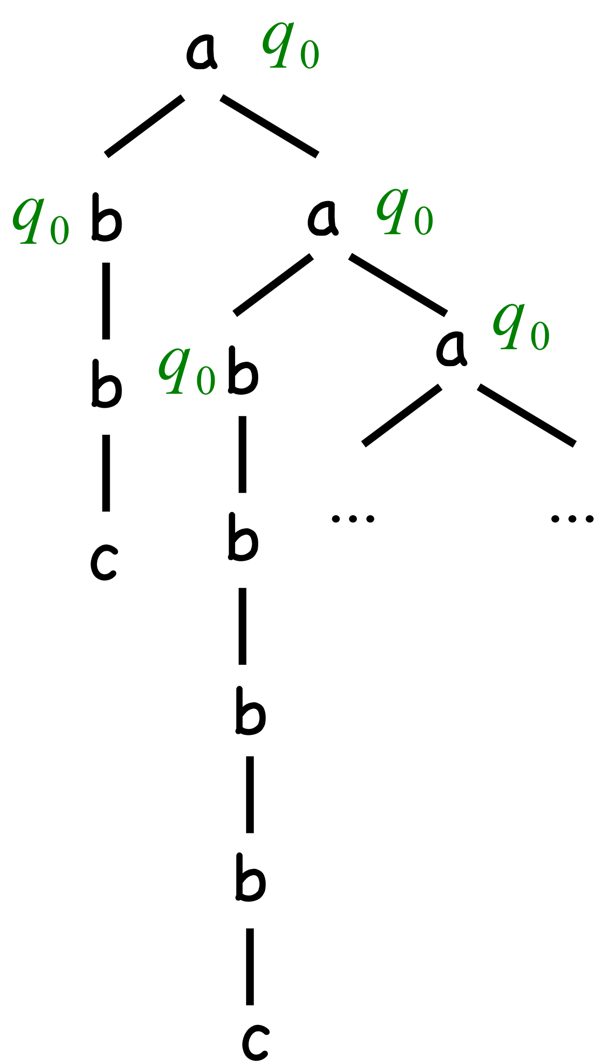
$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Trivial Tree Automata [Klaus A., LMCS07]

(top-down deterministic, all states are final)



$A:$

$\delta(q_0, a) = q_0 q_0$

$\delta(q_1, a) = q_1 q_1$

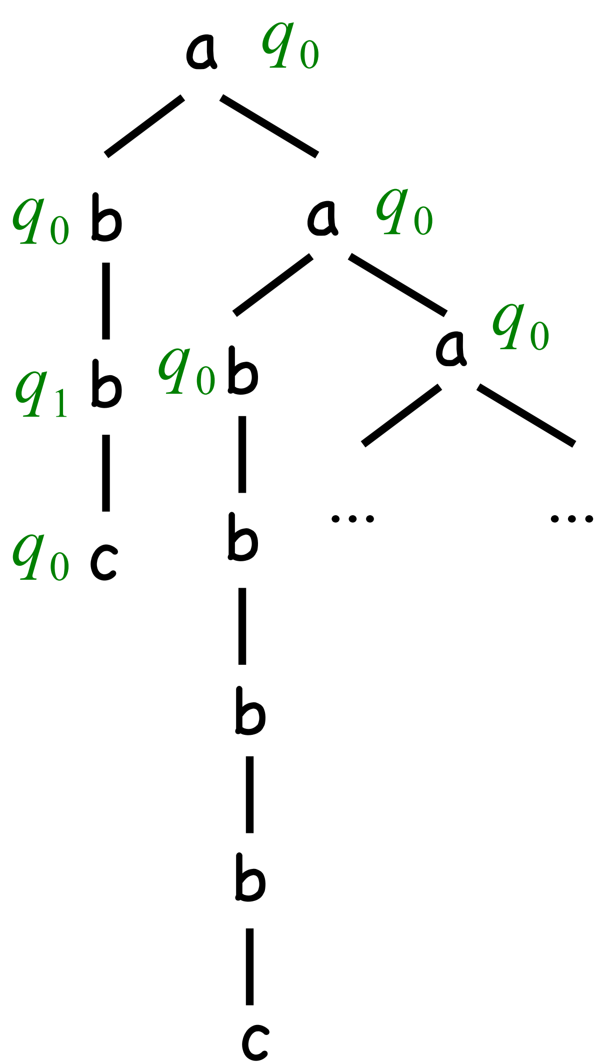
$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Trivial Tree Automata [Klaus A., LMCS07]

(top-down deterministic, all states are final)



$A:$

$\delta(q_0, a) = q_0 q_0$

$\delta(q_1, a) = q_1 q_1$

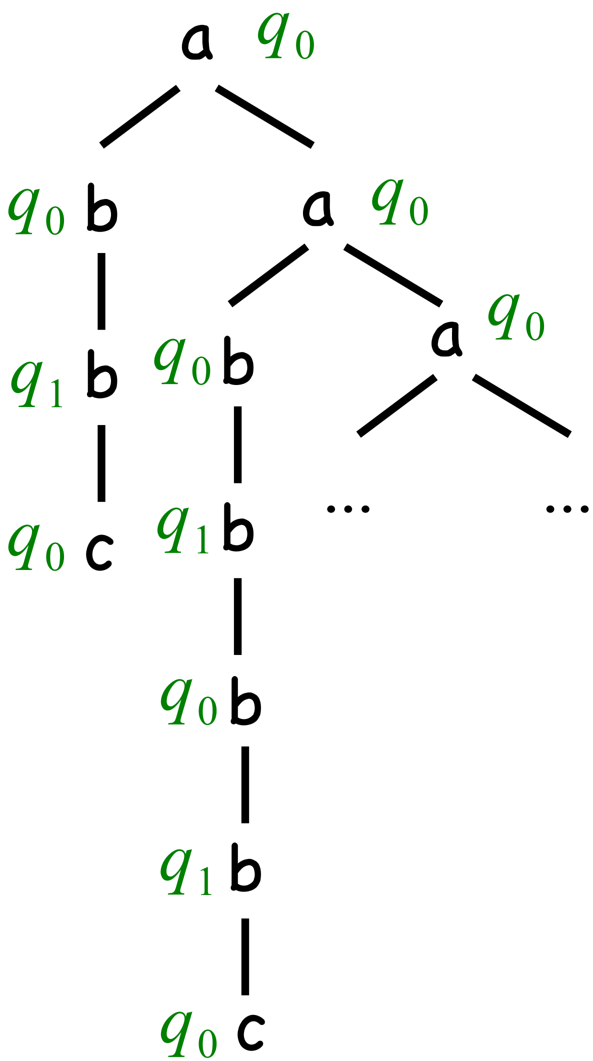
$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Trivial Tree Automata [Klaus A., LMCS07]

(top-down deterministic, all states are final)



$A:$

$\delta(q_0, a) = q_0 q_0$

$\delta(q_1, a) = q_1 q_1$

$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

A accepts Tree(G)

HORS Model Checking

Given G: HORS

A: Alternating parity tree automaton

(formula of modal μ-calculus or MSO logic)

Does A accept Tree(G) ?

Theorem [Ong, LICS06]

HORS model checking is k-EXPTIME-complete for order-k recursion scheme

μHORS Model Checking

Given G: μHORS
A: trivial tree automaton
(for describing safety properties)
Does A accept Tree(G) ?

Theorem [K., Igarashi, ESOP13] [Tsukada, K., FoSSaCS10]
μHORS model checking is undecidable

(Sound and incomplete procedure is concerned)

Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Illustrate abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and onclusion

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

Will files be properly closed?

(Example borrowed/modified from Kobayashi's talk@ESOP13)

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

→

```
G:
FileRd k -> k (Read Close Fun).
Read this k -> r k.
Close this k -> c k.
Fun this k -> br (2@this this k)
            (1@this this (Fun this k)).
S -> FileRd (λx. 3@x x end).
```

A:
Each path of Tree(G) ends with "c"

Will files be properly closed?

Does A accept Tree(G) ?

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

An object is as a tuple of methods

```
FileRd k -> k (Read Close Fun).
Read this k -> r k.
Close this k -> c k.
Fun this k -> br (2@this this k)
             (1@this this (Fun this k)).
S -> FileRd ( λx. 3@x x end).
```

Will files be properly closed?

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

An object is as a tuple of methods

```
FileRd k -> k (Read Close Fun).
Read this k -> r k.
Close this k -> c k.
Fun this k -> br (2@this this k)
              (1@this this (Fun this k)).
S -> F      ( λx. 3@x x end).
```

The implicit parameter "this" in OO
A shorthand for (Read Close Fun)

Will files be properly closed?

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

→

An object is as a tuple of methods

```
FileRd k -> k (Read Close Fun).
Read this k -> r k.
Close this k -> c k.
Fun this k -> br (2@this this k)
                 (1@this this (Fun this k)).
S -> FileRd (λx. 3@x x end).
```

i@x: the i-th projection of x

Will files be properly closed?

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```

```
FileRd k -> k (Read Close Fun).
Read this k -> r k.
Close this k -> c k.
Fun this k -> br (2@this this k)
               (1@this this (Fun this k)).
S -> FileRd (λx. 3@x x end).
```

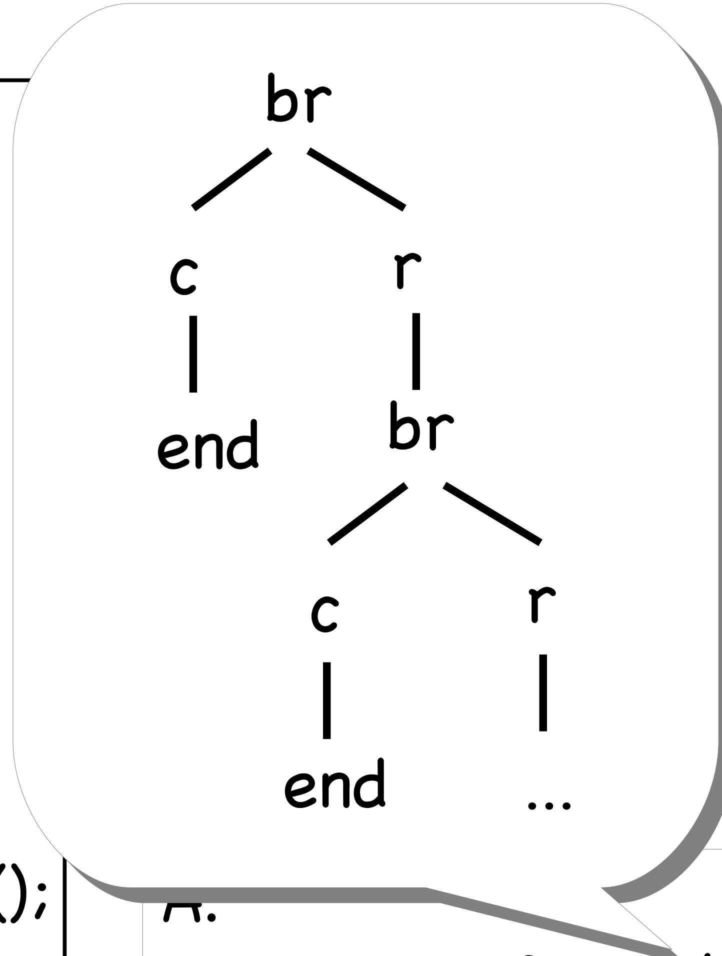
CPS transformation to model event sequences

Will files be properly closed?

Example: Application to OO Verification

```
Class FileRd {
  FileRd(File f) {...}
  read() {...}
  close() {...}
  fun() {
    if * then close()
    else { read(); fun(); }}
}

new FileRd("test.ml").fun();
```



(Read Close Fun).
.
@this this k)
this (Fun this k)).
x. 3@x x end).

A:
Each path of Tree(G) ends with "c"

Will files be properly closed?

Does A accept Tree(G) ? (yes)

Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Illustrate abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and conclusion

Configuration Graph [K., JACM13]

(a product of the reduction of G and A)

$G:$

$S \rightarrow F\,F\,b$ $\qquad B\,h\,x \rightarrow b(h\,x)$

$F\,f\,g \rightarrow a(g(g\,c))(f\,f(B\,g))$

$A:$

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Configuration Graph [K., JACM13]

(a product of the reduction of G and A)

(S, q_0)

↓ [S]

(FFb, q_0)

G:

$S \rightarrow F\,F\,b \qquad B\,h\,x \rightarrow b(h\,x)$

$F\,f\,g \rightarrow a(g(g\,c))(f\,f(B\,g))$

A:

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Configuration Graph [K., JACM13]
(a product of the reduction of G and A)

(S, q_0)

↓ [S]

(FFb, q_0)

(t, q): the term tree of t is to be accepted by A from q

G:

$S \rightarrow F\, F\, b$ $\qquad B\, h\, x \rightarrow b(h\, x)$

$F\, f\, g \rightarrow a(g(g\, c))(f\, f(B\, g))$

A:

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Configuration Graph [K., JACM13]
(a product of the reduction of G and A)

(S, q_0)

↓ [S]

$(FFb,\ q_0)$

↓ [F]

$(a(b(bc))\ (FF(Bb)),\ q_0)$

(t, q): the term tree of t is to be accepted by A from q

$G:$

$$S \to F\,F\,b \qquad B\,h\,x \to b(h\,x)$$

$$F\,f\,g \to a(g(g\,c))(f\,f(B\,g))$$

$A:$

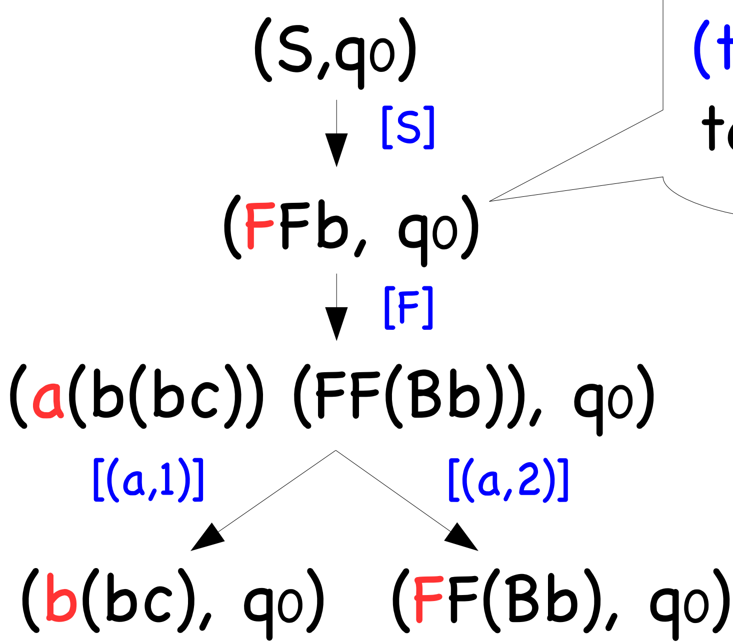
$$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$$

$$\delta(q_0, c) = \epsilon$$

Configuration Graph [K., JACM13]

(a product of the reduction of G and A)



(t, q): the term tree of t is to be accepted by A from q

$G:$

$S \rightarrow F\,F\,b \qquad B\,h\,x \rightarrow b(h\,x)$

$F\,f\,g \rightarrow a(g(g\,c))(f\,f(B\,g))$

$A:$

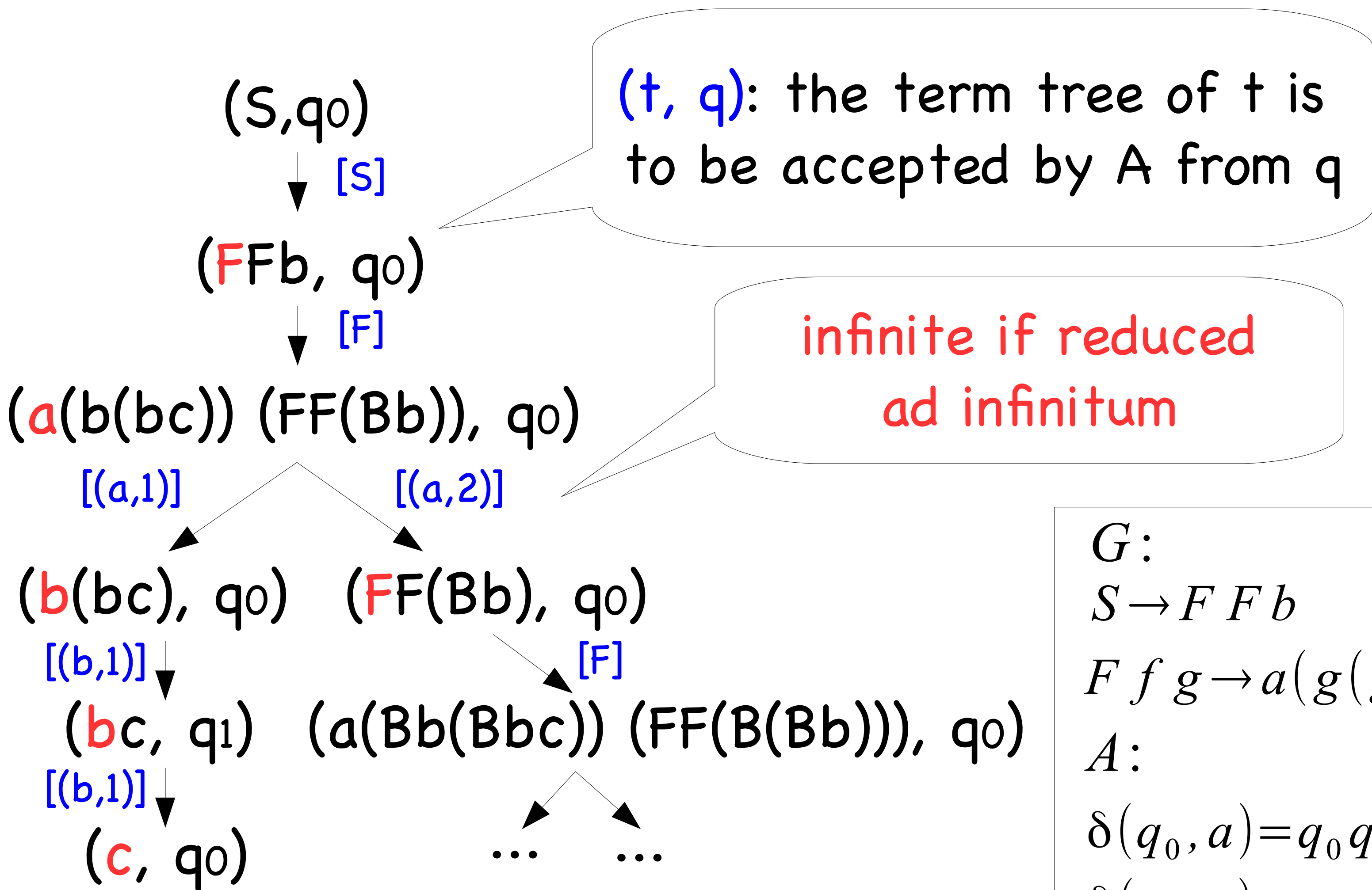
$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Configuration Graph [K., JACM13]

(a product of the reduction of G and A)



$G:$

$S \to F\,F\,b \qquad B\,h\,x \to b(h\,x)$

$F\,f\,g \to a(g(g\,c))(f\,f(B\,g))$

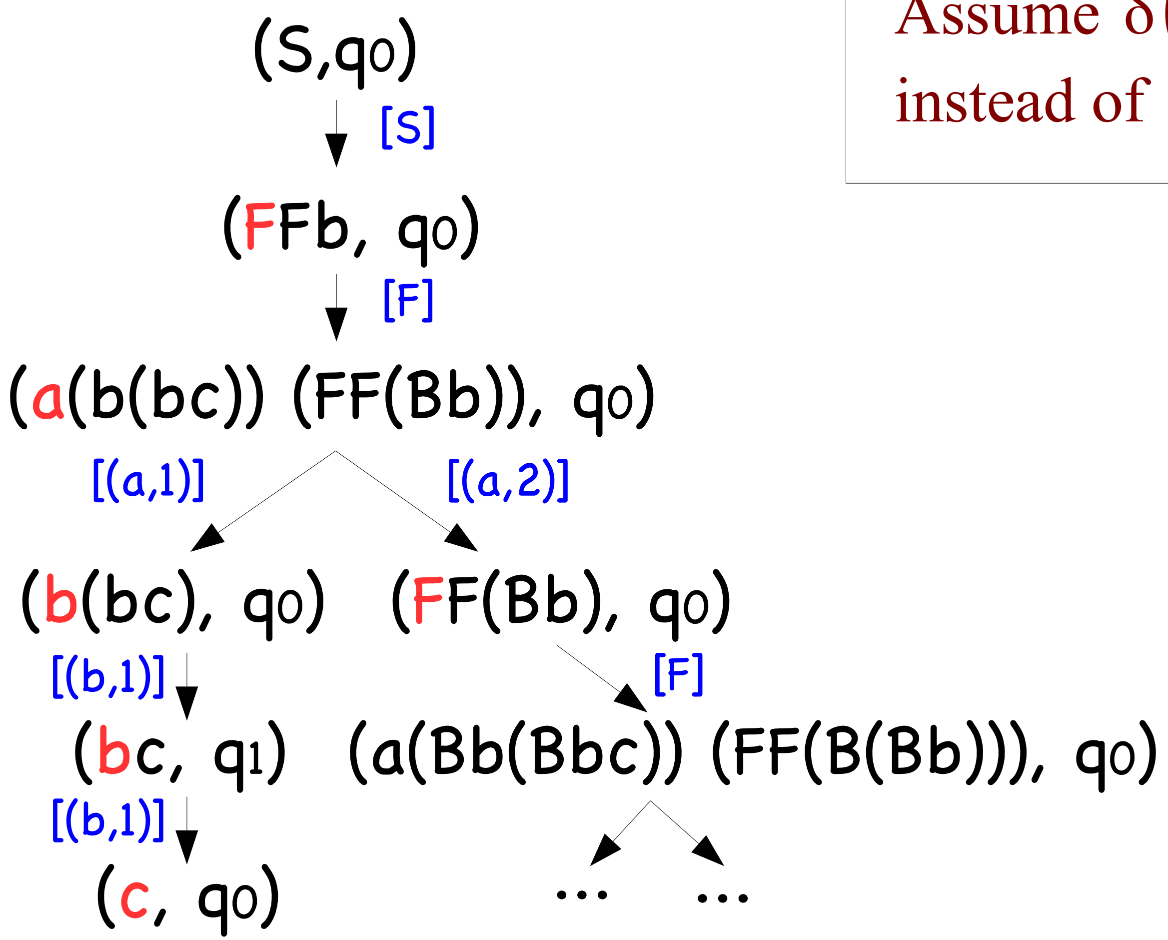
$A:$

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Configuration Graph [K., JACM13]
(a product of the reduction of G and A)



Assume $\delta(q_1, c) = \epsilon$
instead of $\delta(q_0, c) = \epsilon$

G:

$S \to F\,F\,b \qquad B\,h\,x \to b(h\,x)$

$F\,f\,g \to a(g(g\,c))(f\,f(B\,g))$

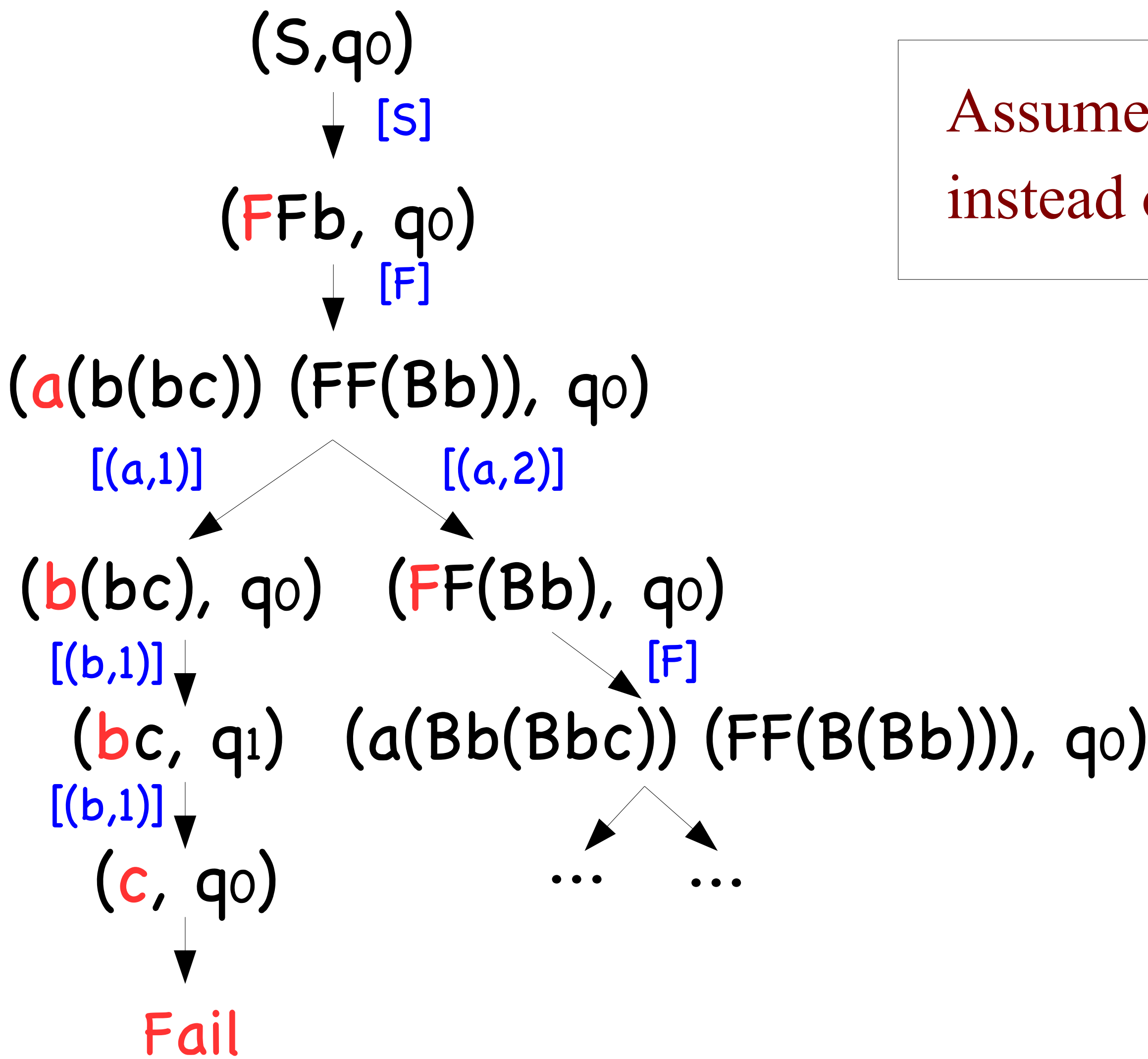
A:

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_1, c) = \epsilon$

Configuration Graph [K., JACM13]
(a product of the reduction of G and A)



Assume $\delta(q_1, c) = \epsilon$
instead of $\delta(q_0, c) = \epsilon$

$G:$

$S \to F\,F\,b \qquad B\,h\,x \to b(h\,x)$

$F\,f\,g \to a(g(g\,c))(f\,f(B\,g))$

$A:$

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_1, c) = \epsilon$

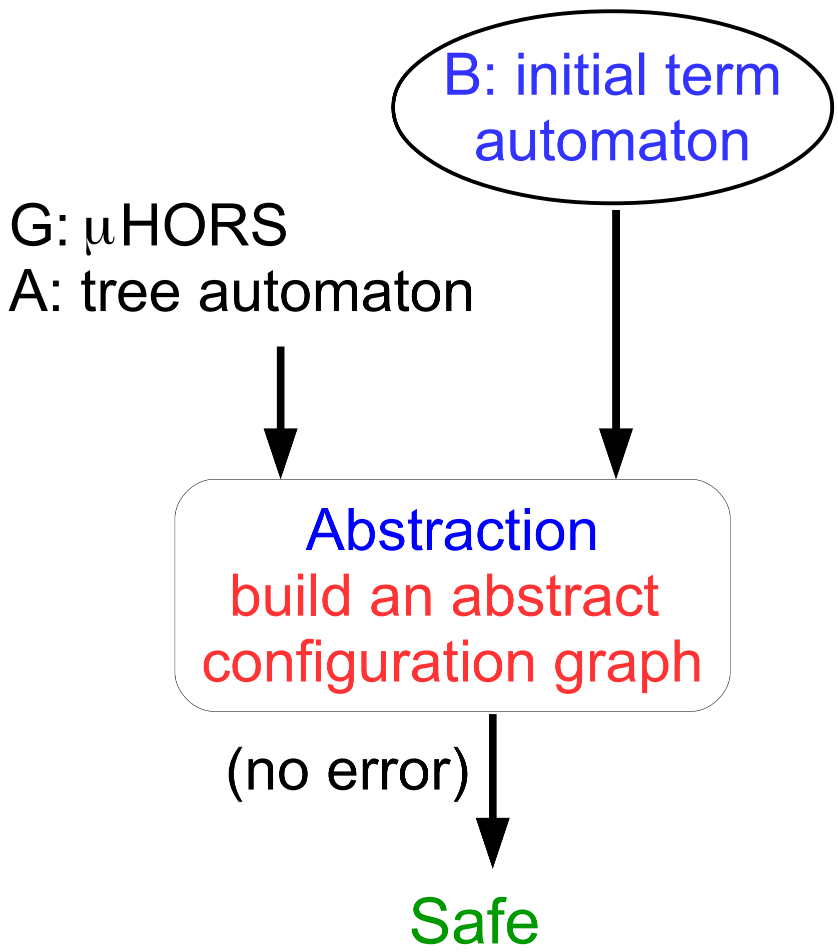
Restate μHORS model checking

Given G: μHORS
A: trivial tree automaton
(for describing safety properties)
Does configuration graph for G and A contain Fail?

Theorem
A accepts Tree(G) <=>
Configuration graph for G and A does not contain Fail

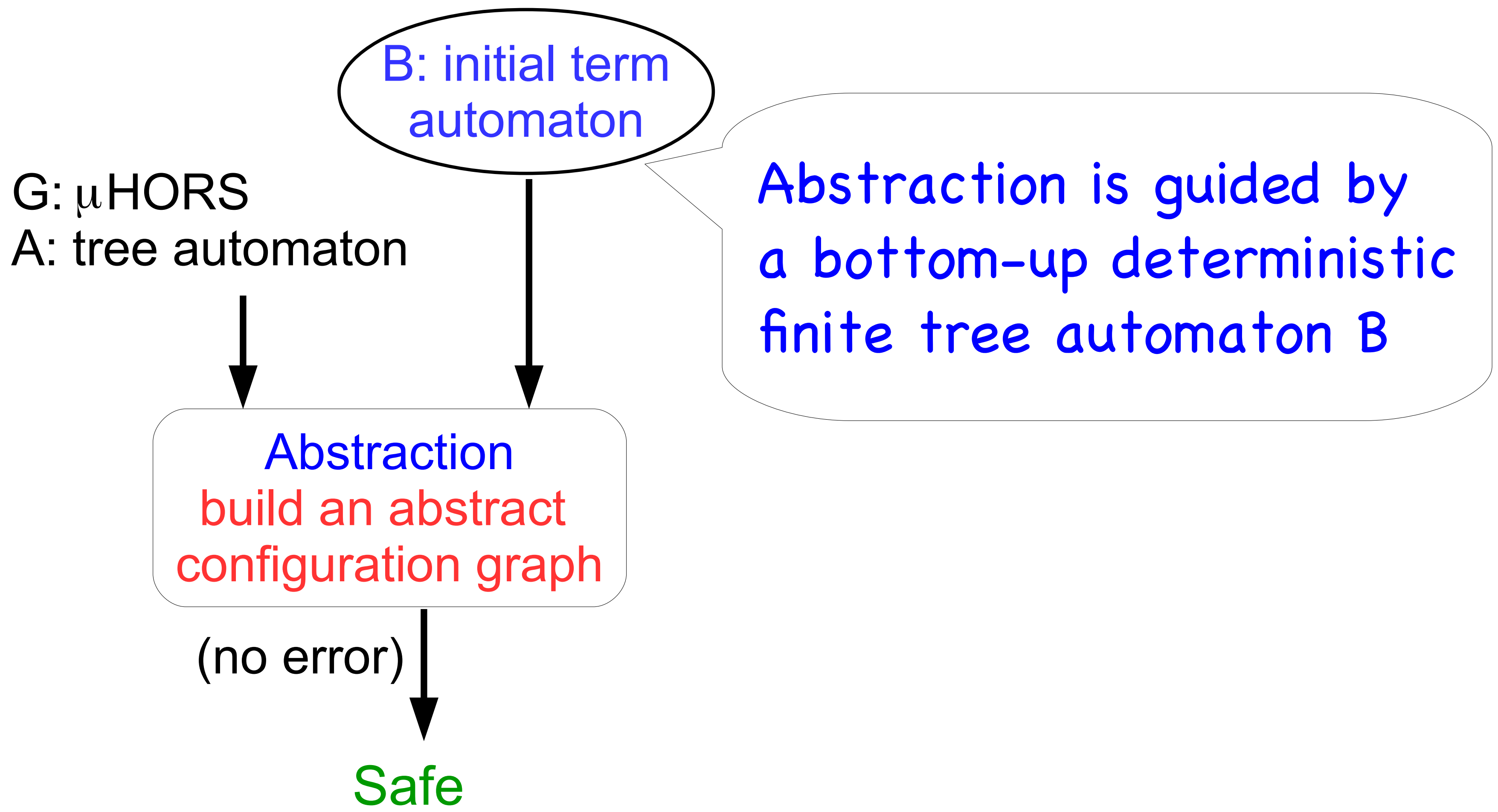
Our Approach: Overview

Construct a finite abstract configuration graph for approximating all reduction sequences of G and A



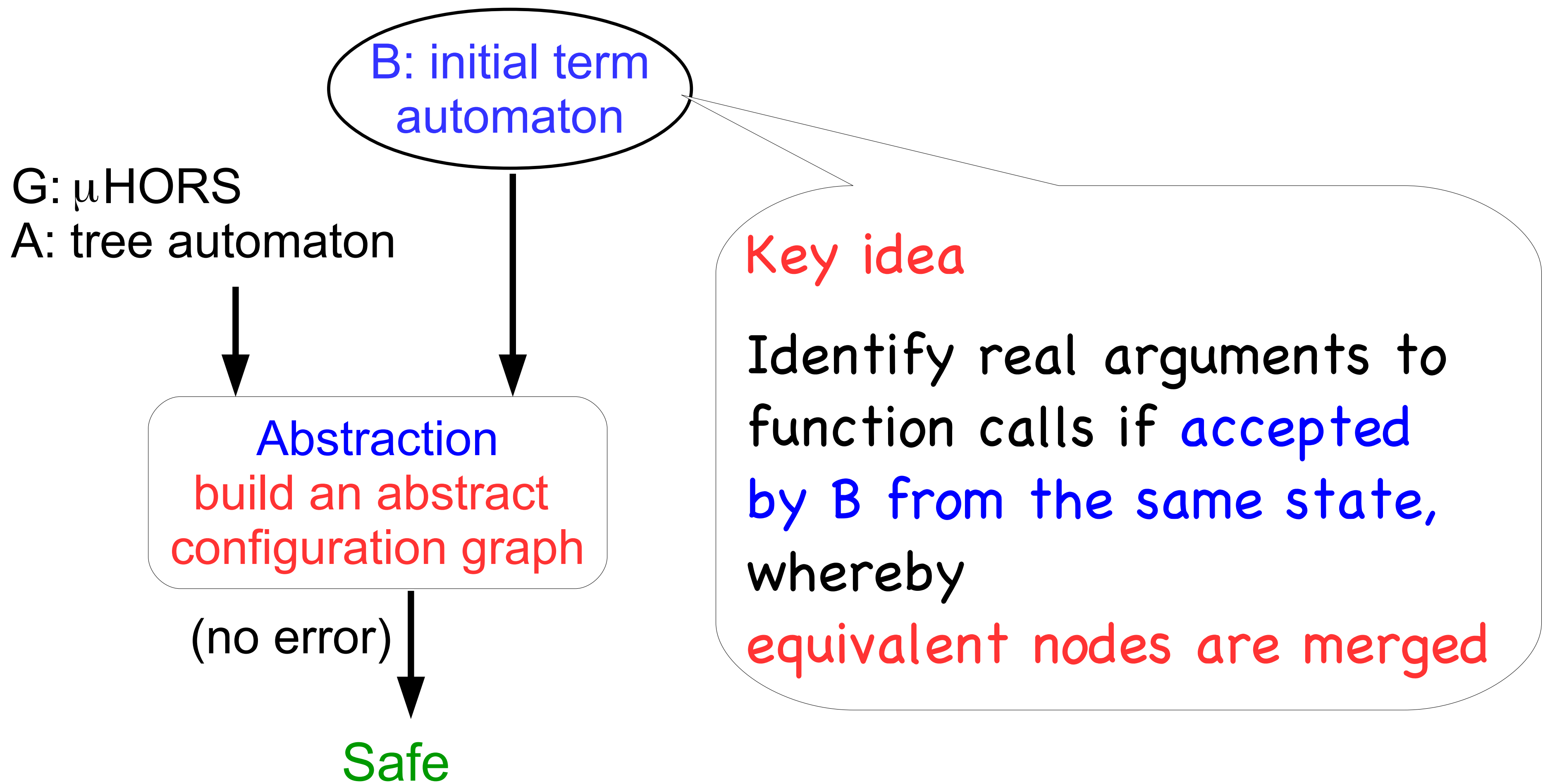
Our Approach: Overview

Construct a finite abstract configuration graph for approximating all reduction sequences of G and A



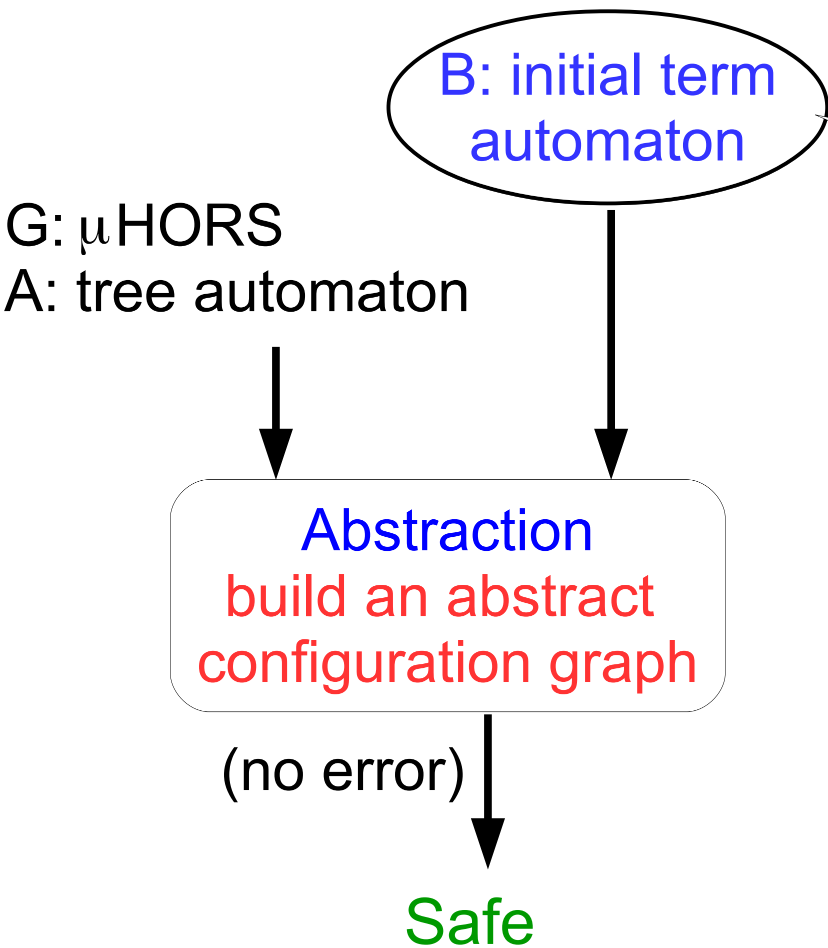
Our Approach: Overview

Construct a finite abstract configuration graph for approximating all reduction sequences of G and A



Our Approach: Overview

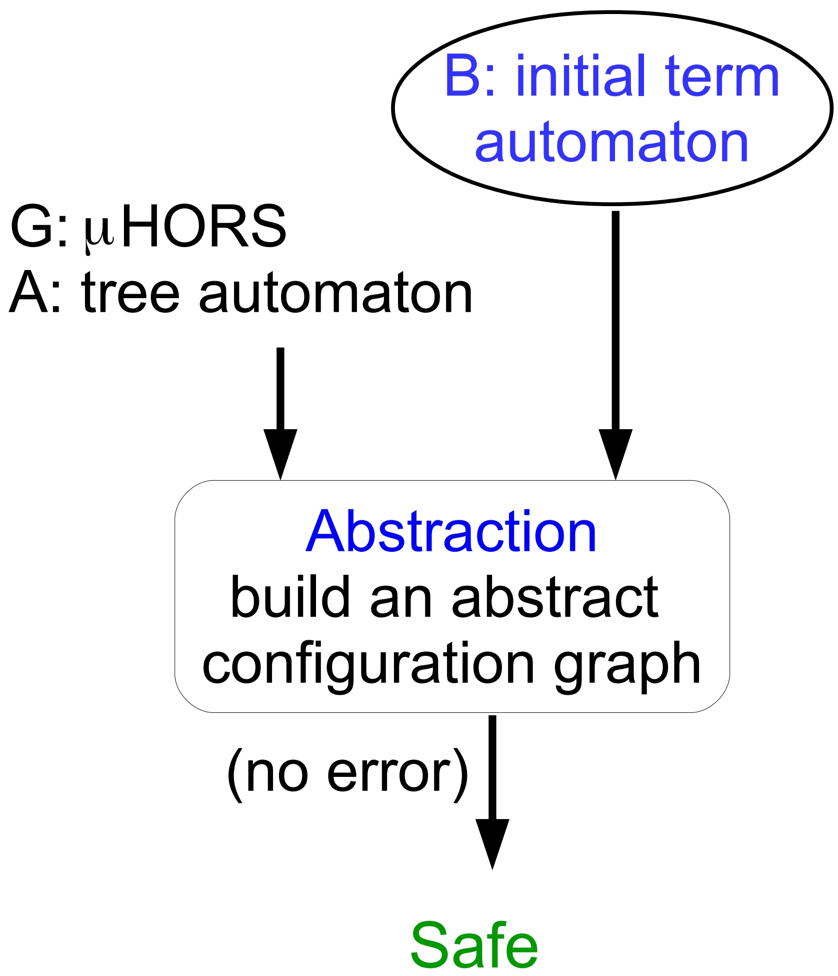
Construct a finite abstract configuration graph for approximating all reduction sequences of G and A



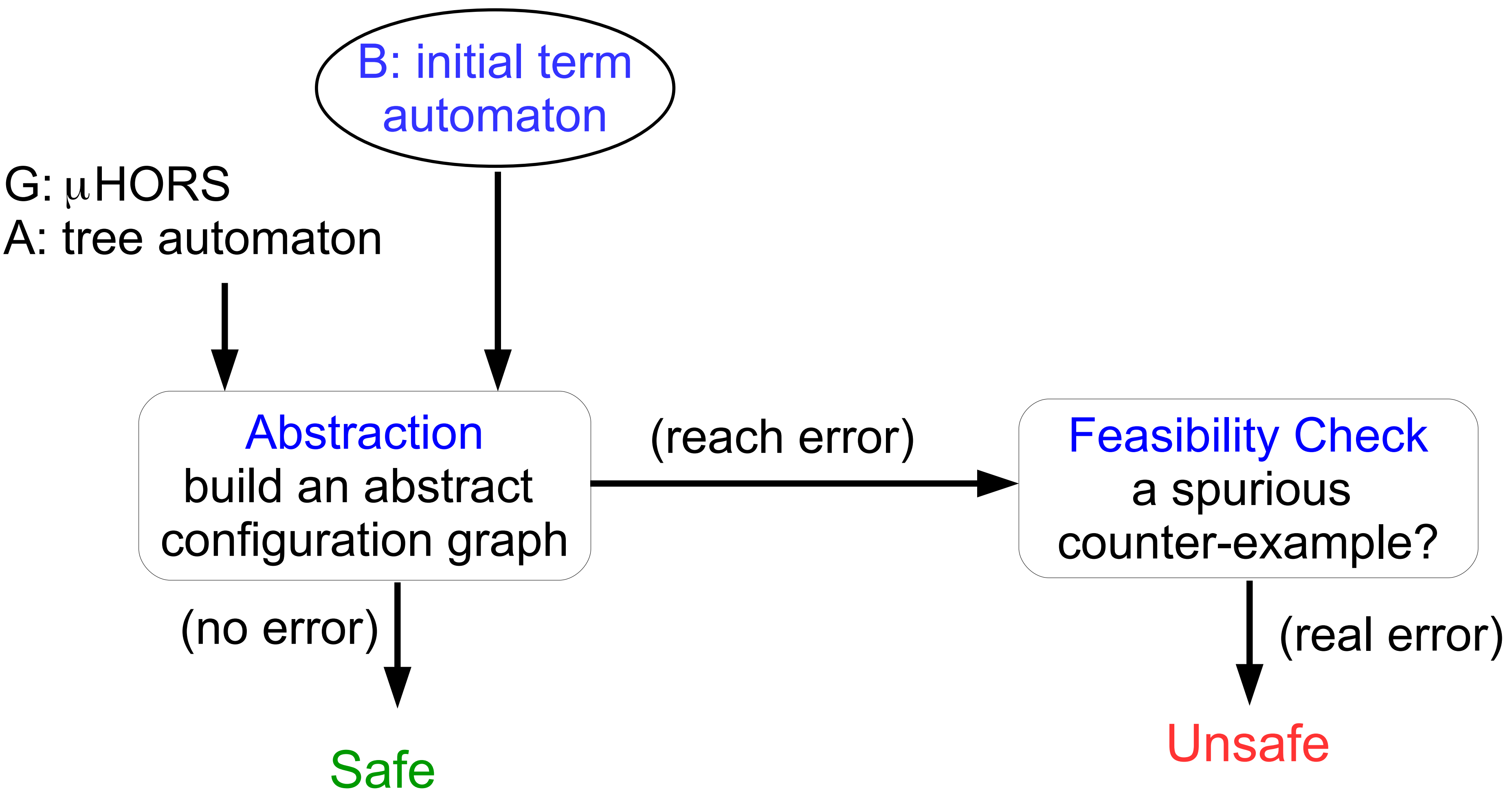
Inspiration [Ramsay+, POPL14]

Identify real arguments to function calls if inhabit the same intersection types, whereby equivalent nodes are merged

Our Approach: Overview

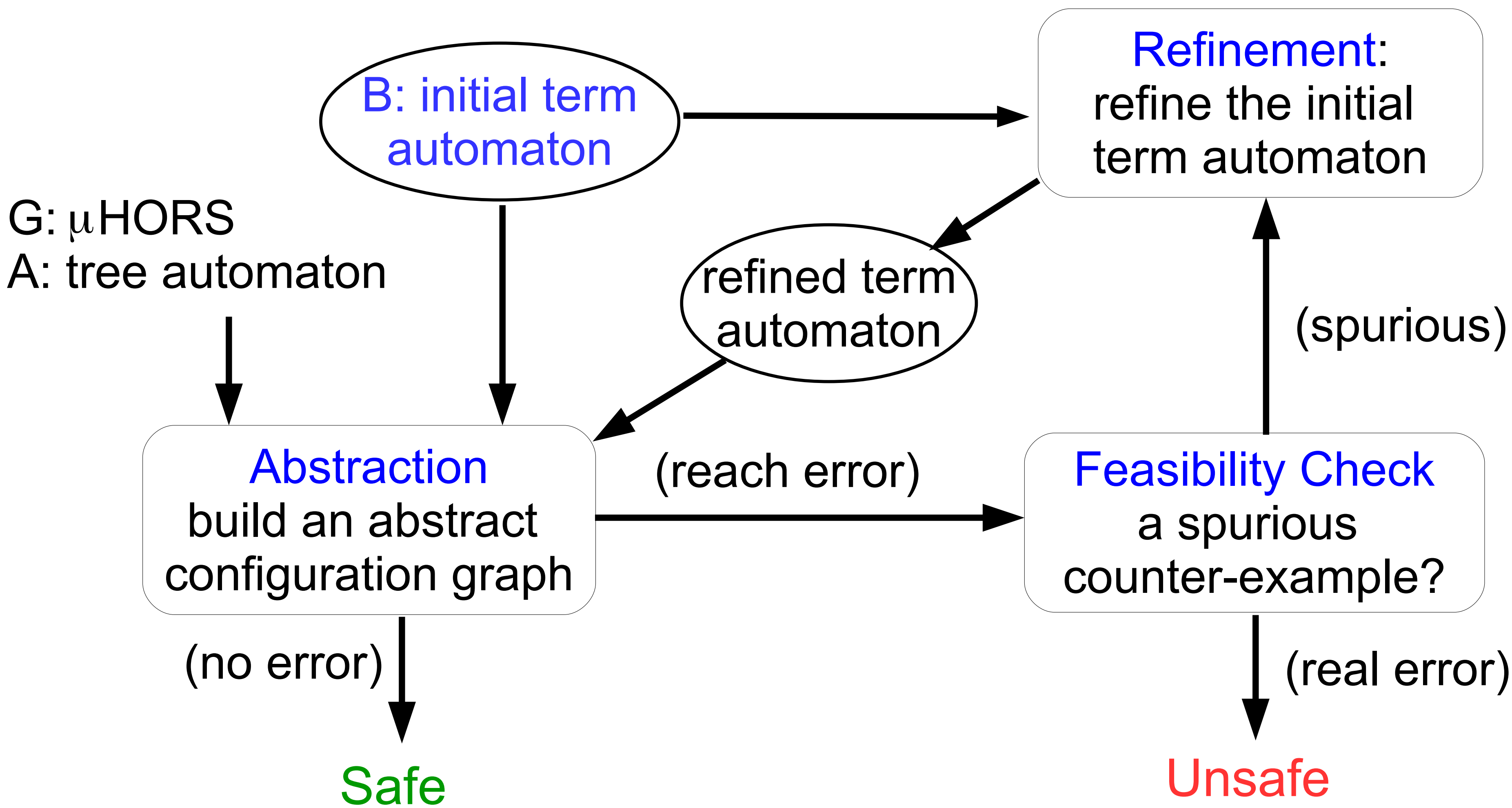


Our Approach: Overview



Our Approach: Overview

Refine abstraction by cloning states and transitions of B from which extract a refined automaton



Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Illustrating abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and conclusion

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(\color{darkred}{F}\; F\; b, q_0)$$

$G:$

$S \rightarrow F\;F\;b$ $\quad$ $B\;h\;x \rightarrow b(h\;x)$

$F\;f\;g \rightarrow a(g(g\;c))(f\;f(B\;g))$

$A:$

$\delta(q_0, a) = q_0 q_0$ $\quad$ $\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1$ $\quad$ $\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(F\ F\ b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1\ f^1(B\ g^2)), q_0)$$

Bindings:

$$g^2 \leftarrow b$$

$$f^1 \leftarrow F$$

$G:$

$S \rightarrow F\ F\ b$

$B\ h\ x \rightarrow b(h\ x)$

$F\ f\ g \rightarrow a(g(g\ c))(f\ f(B\ g))$

$A:$

$\delta(q_0, a) = q_0 q_0$

$\delta(q_1, a) = q_1 q_1$

$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(F\, F\, b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1\, f^1(B\, g^2)), q_0)$$

$$\swarrow \qquad \searrow$$

$$(g^2(g^2 c), q_0) \qquad (f^1\, f^1(B g^2), q_0)$$

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$

$G:$

$$S \rightarrow F\, F\, b \qquad B\, h\, x \rightarrow b(h\, x)$$

$$F\, f\, g \rightarrow a(g(g\, c))(f\, f(B\, g))$$

$A:$

$$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$$

$$\delta(q_0, c) = \epsilon$$

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(F\ F\ b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1 f^1(B\ g^2)), q_0)$$

$$\swarrow \qquad \searrow$$

$$(g^2(g^2 c), q_0) \qquad (f^1 f^1(Bg^2), q_0)$$

$$\downarrow \qquad\qquad \downarrow$$

$$(b(g^2 c), q_0) \qquad (F\ f^1(Bg^2), q_0)$$

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$

G:

$S \rightarrow F\ F\ b \qquad B\ h\ x \rightarrow b(h\ x)$

$F\ f\ g \rightarrow a(g(g\ c))(f\ f(B\ g))$

A:

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(F\ F\ b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1 f^1(B g^2)), q_0)$$

$$\swarrow \qquad \searrow$$

$$(g^2(g^2 c), q_0) \qquad (f^1 f^1(B g^2), q_0)$$

$$\downarrow \qquad \downarrow$$

$$(b(g^2 c), q_0) \qquad (F f^1(B g^2), q_0)$$

$$\downarrow$$

$$(a(g^4(g^4 c))(f^3 f^3(B g^4)), q_0)$$

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$
$$f^3 \leftarrow F$$
$$g^4 \leftarrow B\, g^2$$

$G:$

$S \rightarrow F\ F\ b \qquad B\ h\ x \rightarrow b(h\ x)$

$F\ f\ g \rightarrow a(g(g\ c))(f\ f(B\ g))$

$A:$

$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

Given $\delta_B(Bb) = \delta_B(b)$

$$(S, q_0)$$

$$\downarrow$$

$$(F\ F\ b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1\ f^1(B\ g^2)), q_0)$$

$$\swarrow \qquad \searrow$$

$$(g^2(g^2 c), q_0) \qquad (f^1\ f^1(Bg^2), q_0)$$

$$\downarrow \qquad\qquad \downarrow$$

$$(b(g^2 c), q_0) \qquad (F\ f^1(Bg^2), q_0)$$

$$\downarrow$$

$$(a(g^4(g^4 c))(f^3\ f^3(B\ g^4)), q_0)$$

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$
$$f^3 \leftarrow F$$
$$g^4 \leftarrow B\ g^2$$

G:

$$S \to F\ F\ b \qquad B\ h\ x \to b(h\ x)$$

$$F\ f\ g \to a(g(g\ c))(f\ f(B\ g))$$

A:

$$\delta(q_0, a) = q_0 q_0 \qquad \delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_1 q_1 \qquad \delta(q_1, b) = q_0$$

$$\delta(q_0, c) = \epsilon$$

Example: Abstraction

(S, q_0)

↓

$(F\, F\, b, q_0)$

↓

$(a(g^2(g^2 c))(f^1 f^1(B g^2)), q_0)$

$(g^2(g^2 c), q_0)$ $\quad$ $(f^1 f^1(Bg^2$ …

↓ $\quad$ ↓

$(b(g^2 c), q_0)$ $\quad$ $(F\, f^1(Bg$ … $), q_0)$

↓

$(a(g^4(g^4 c))(f^3 f^3(B g^4)), q_0)$

Given $\delta_B(Bb) = \delta_B(b)$

Bindings:

$g^2 \leftarrow b$

$f^1 \leftarrow F$

$f^3 \leftarrow F$

$g^4 \leftarrow B\, g^2$

Do not distinguish g^2 and g^4:

g^2 is bound with b

g^4 is bound with Bb

… $x \rightarrow b(h\, x)$

… $f(B\, g))$

A:

$\delta(q_0, a) = q_0 q_0$ $\quad$ $\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1$ $\quad$ $\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

$$(S, q_0)$$

$$\downarrow$$

$$(F\ F\ b, q_0)$$

$$\downarrow$$

$$(a(g^2(g^2 c))(f^1 f^1(B\ g^2)), q_0)$$

$$(g^2(g^2 c), q_0) \qquad (f^1 f^1(Bg^2), q_0)$$

$$\downarrow \qquad\qquad \downarrow$$

$$(b(g^2 c), q_0) \qquad (F\ f^1(Bg^2), q_0)$$

$$\downarrow$$

$$(a(g^4(g^4 c))(f^3 f^3(B\ g^4)), q_0)$$

Given $\delta_B(Bb) = \delta_B(b)$

identify the two nodes

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$
$$f^3 \leftarrow F$$
$$g^4 \leftarrow B\ g^2$$

G:

$S \rightarrow F\ F\ b$

$B\ h\ x \rightarrow b(h\ x)$

$F\ f\ g \rightarrow a(g(g\ c))(f\ f(B\ g))$

A:

$\delta(q_0, a) = q_0 q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction

Given $\delta_B(Bb) = \delta_B(b)$

$$(S, q_0)$$

$$(F\,F\,b, q_0)$$

$$(a(g^2(g^2 c))(f^1 f^1(B\,g^2)), q_0)$$

$$(g^2(g^2 c), q_0) \qquad (f^1 f^1(Bg^2), q_0)$$

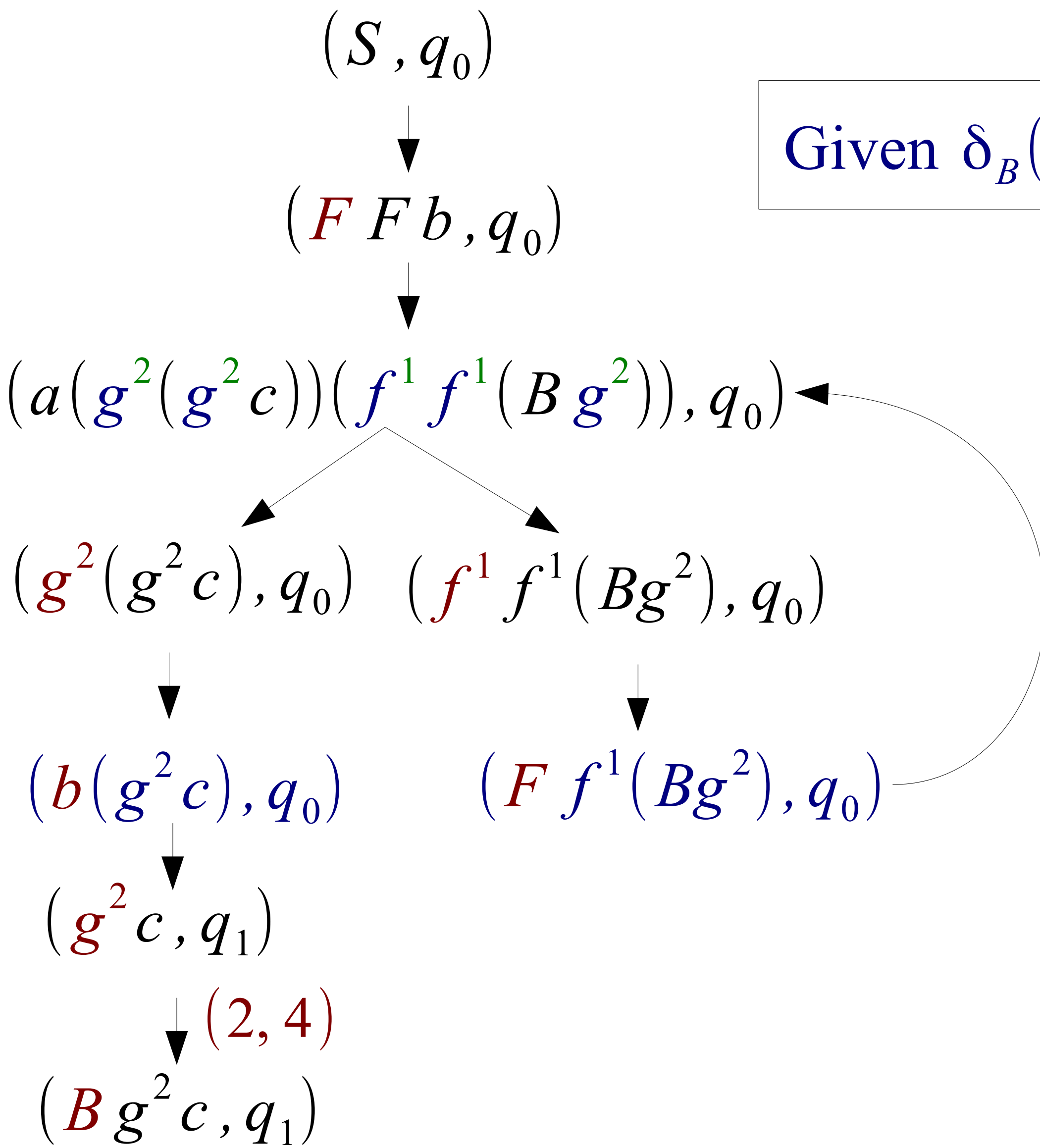
$$(b(g^2 c), q_0) \qquad (F\,f^1(Bg^2), q_0)$$

Bindings:

$$g^2 \leftarrow b$$
$$f^1 \leftarrow F$$
$$f^3 \leftarrow F$$
$$g^4 \leftarrow B\,g^2$$

Example: Abstraction

Given $\delta_B(Bb) = \delta_B(b)$



Bindings:

$g^2 \leftarrow b$

$f^1 \leftarrow F$

$f^3 \leftarrow F$

$g^4 \leftarrow B\, g^2$

$h^5 \leftarrow g^2$

$x^6 \leftarrow c$

$G:$

$S \rightarrow F\ F\ b$

$B\ h\ x \rightarrow b(h\ x)$

$F\ f\ g \rightarrow a(g(g\ c))(f\ f(B\ g))$

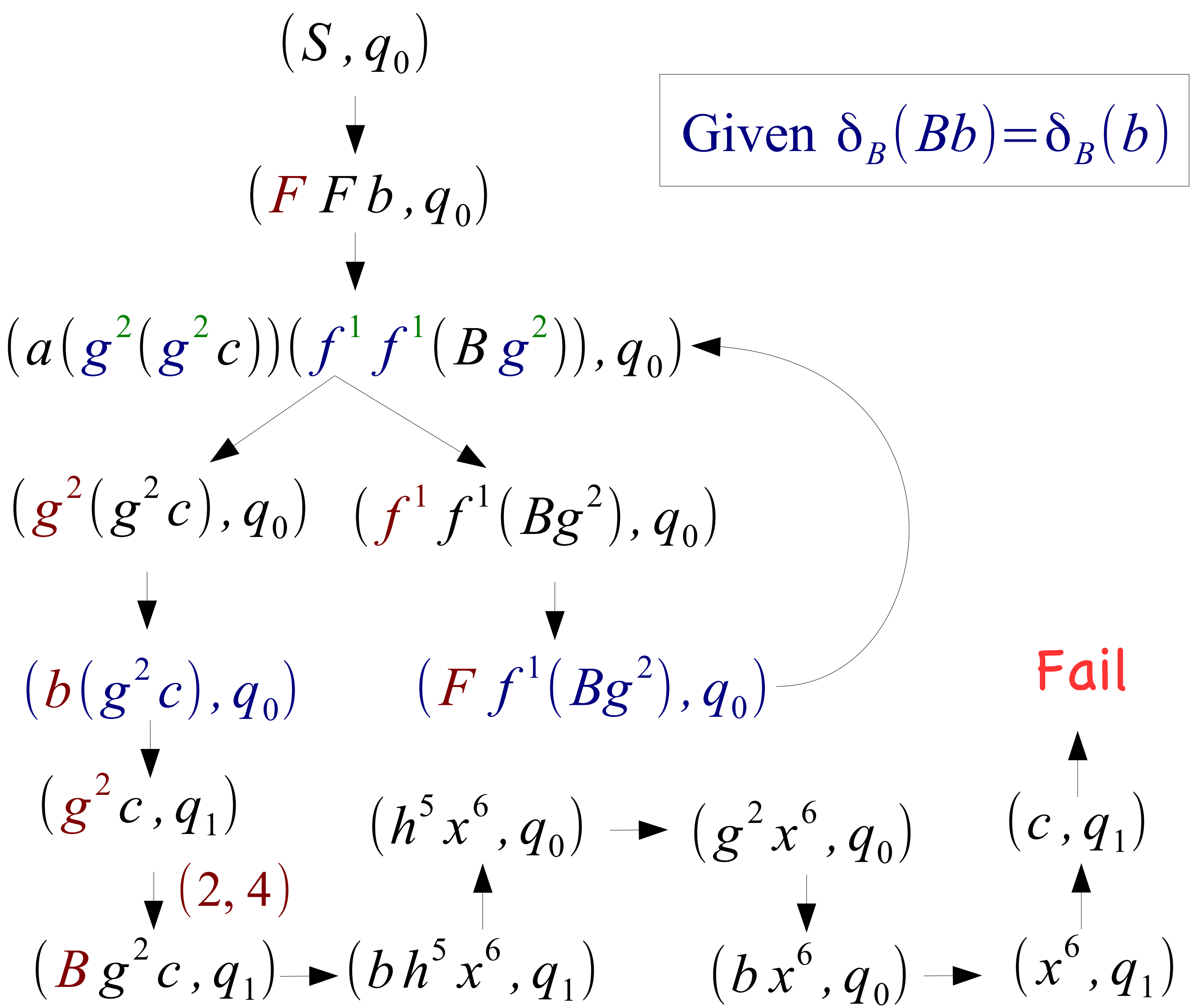
$A:$

$\delta(q_0, a) = q_0 q_0$ $\qquad$ $\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1 q_1$ $\qquad$ $\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

Example: Abstraction



Bindings:

$g^2 \leftarrow b$

$f^1 \leftarrow F$

$f^3 \leftarrow F$

$g^4 \leftarrow B g^2$

$h^5 \leftarrow g^2$

$x^6 \leftarrow c$

A:

$\delta(q_0, a) = q_0 q_0$

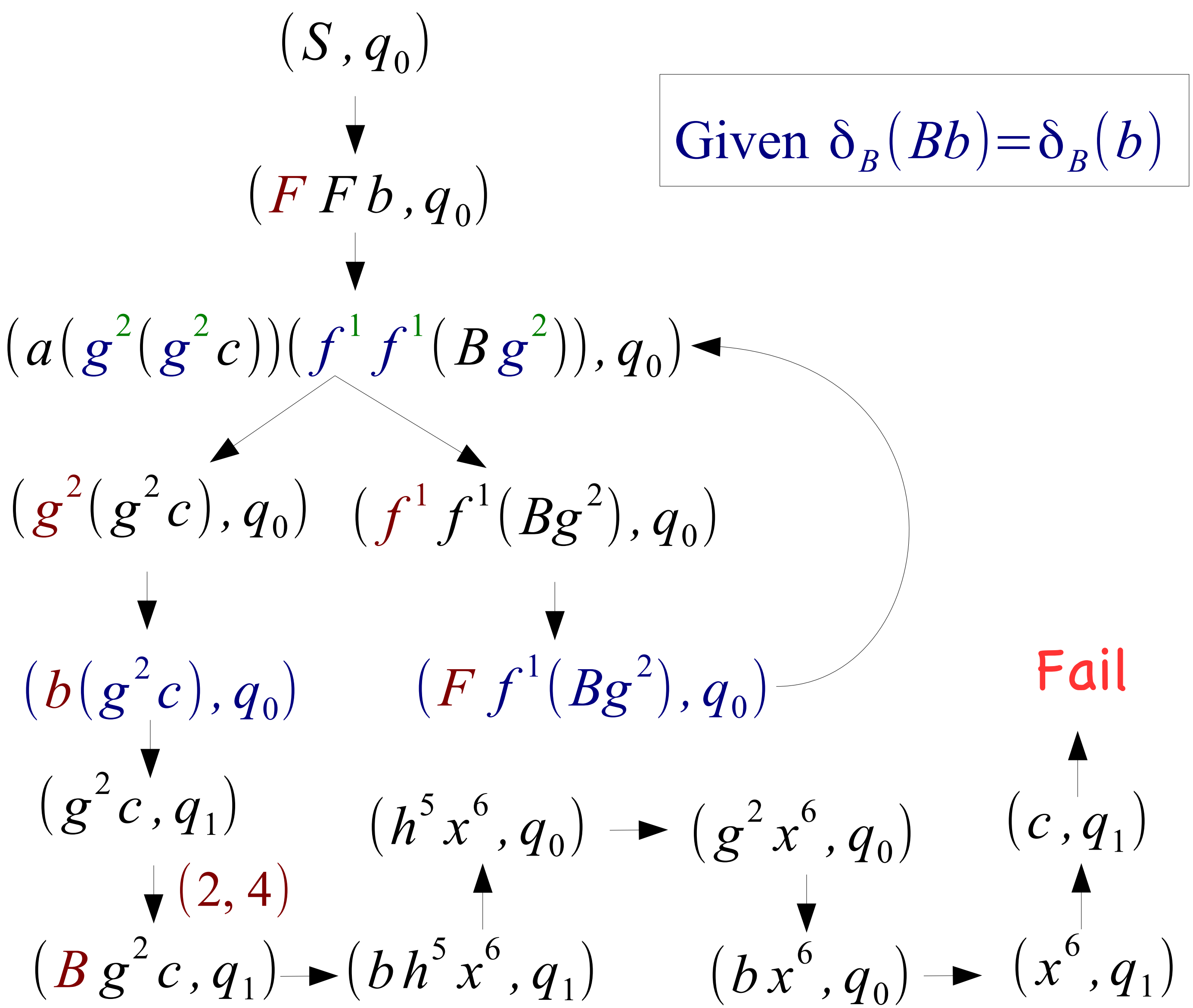
$\delta(q_1, a) = q_1 q_1$

$\delta(q_0, b) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, c) = \epsilon$

How to Eliminate Counterexample



Bindings:

$g^2 \leftarrow b$

$f^1 \leftarrow F$

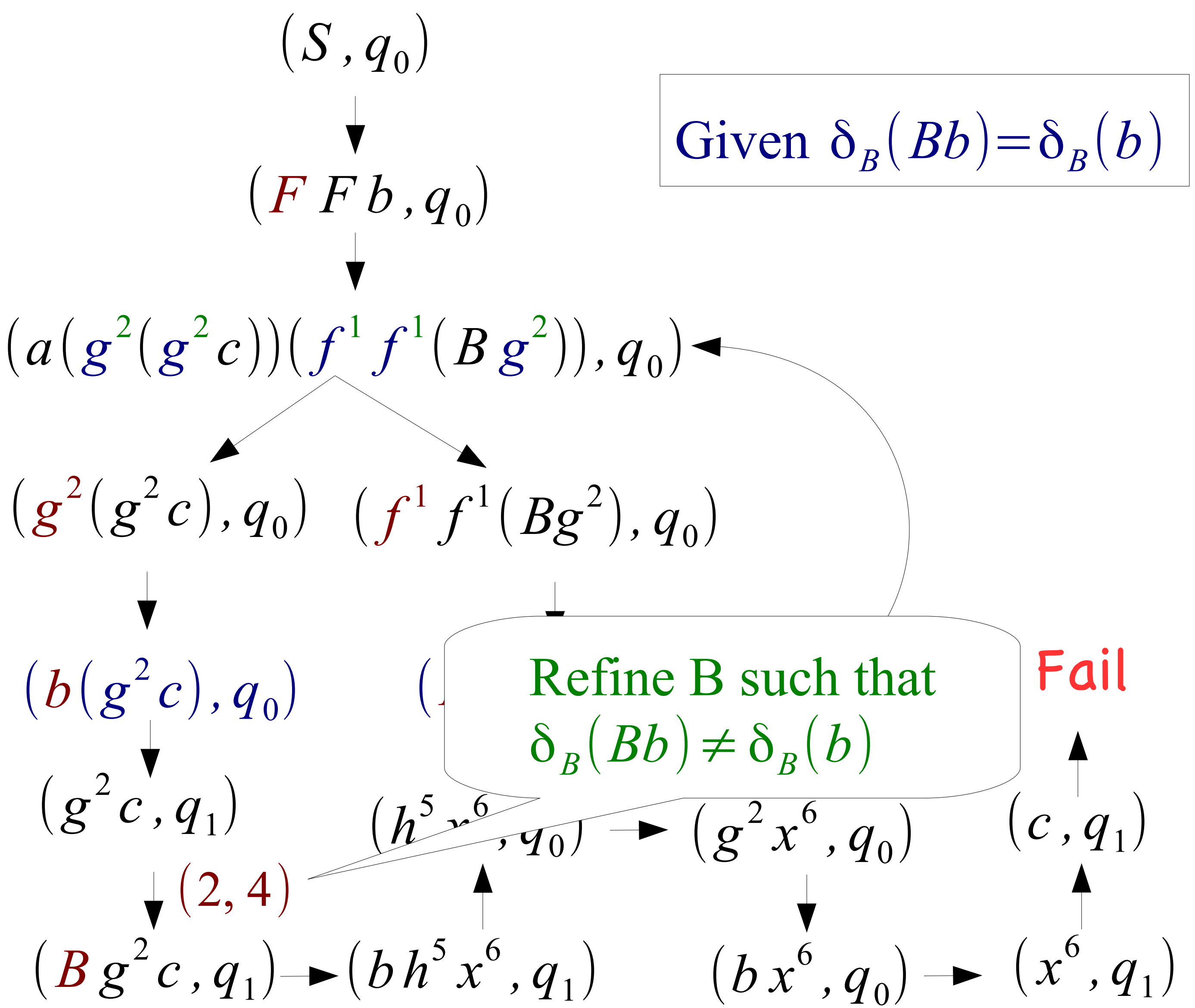
$f^3 \leftarrow F$

$g^4 \leftarrow B\, g^2$

$h^5 \leftarrow g^2$

$x^6 \leftarrow c$

How to Eliminate Counterexample



Bindings:

$g^2 \leftarrow b$

$f^1 \leftarrow F$

$f^3 \leftarrow F$

$g^4 \leftarrow B\,g^2$

$h^5 \leftarrow g^2$

$x^6 \leftarrow c$

How to Refine Abstraction

1. Cloning (k-copies) states and transitions of B
2. Extract a refined B' satisfying the constraint
3. If failed, increase k to k+1 and go to Step 1

How to Refine Abstraction

1. Cloning (k-copies) states and transitions of B
2. Extract a refined B' satisfying the constraint
3. If failed, increase k to k+1 and go to Step 1

↓

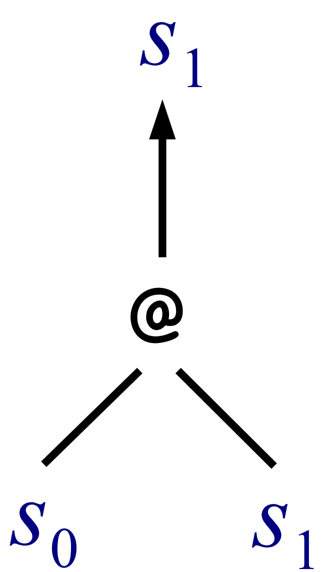
Reduced to SMT Solving for formulas of uninterpreted functions

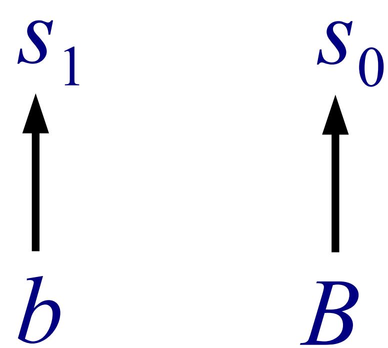
(not really do cloning, it's daunting...)

Example: Abstraction Refinement

$$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$$

term trees are represented using an explicit application symbol "@"

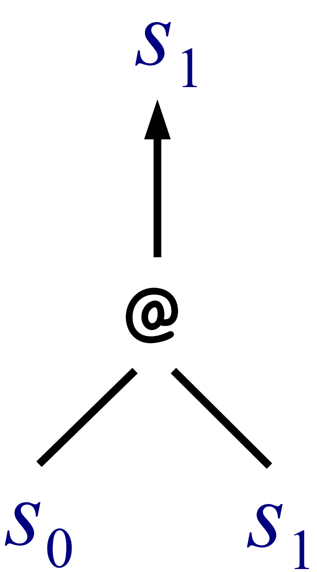


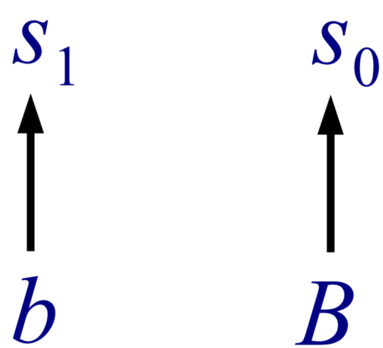


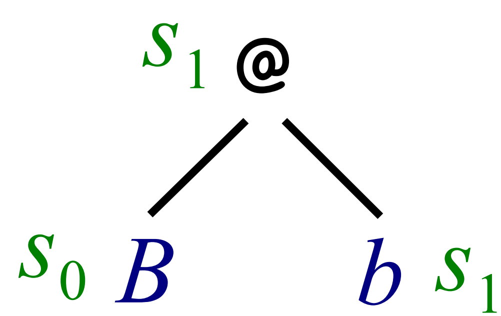
Example: Abstraction Refinement

$$@\ s_0\ s_1 \rightarrow_B s_1 \quad b \rightarrow_B s_1 \quad B \rightarrow_B s_0$$

“Bb” is as “@Bb”



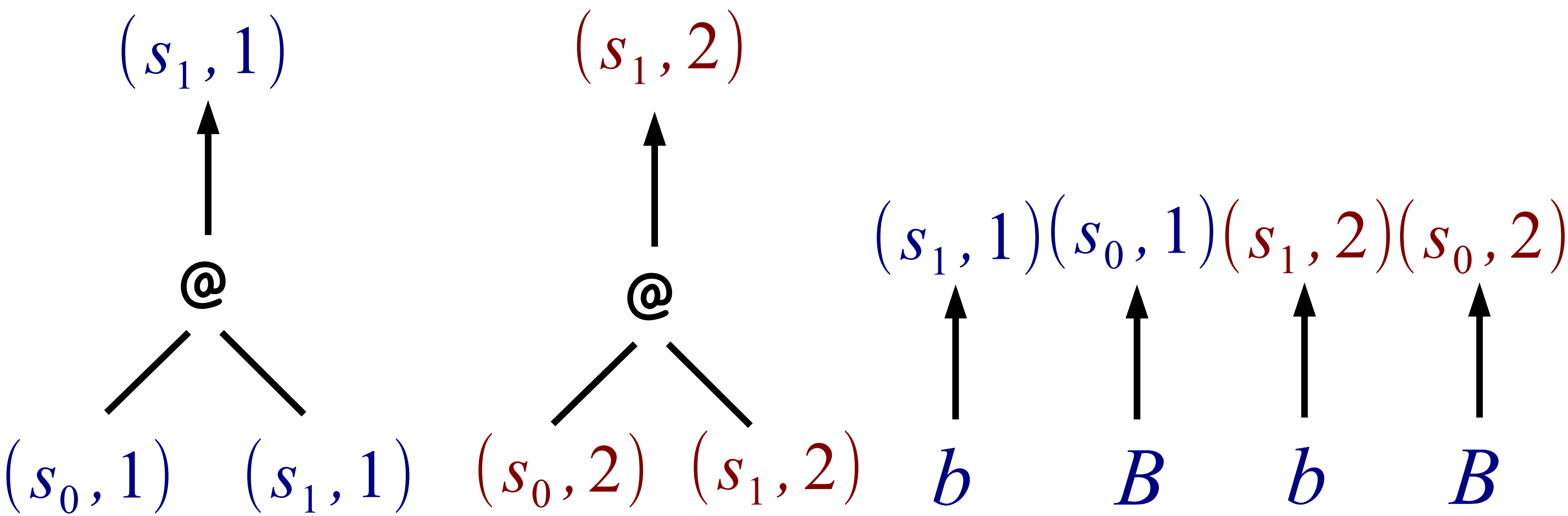




$$\delta_B(@Bb) = s_1$$

Example: Abstraction Refinement

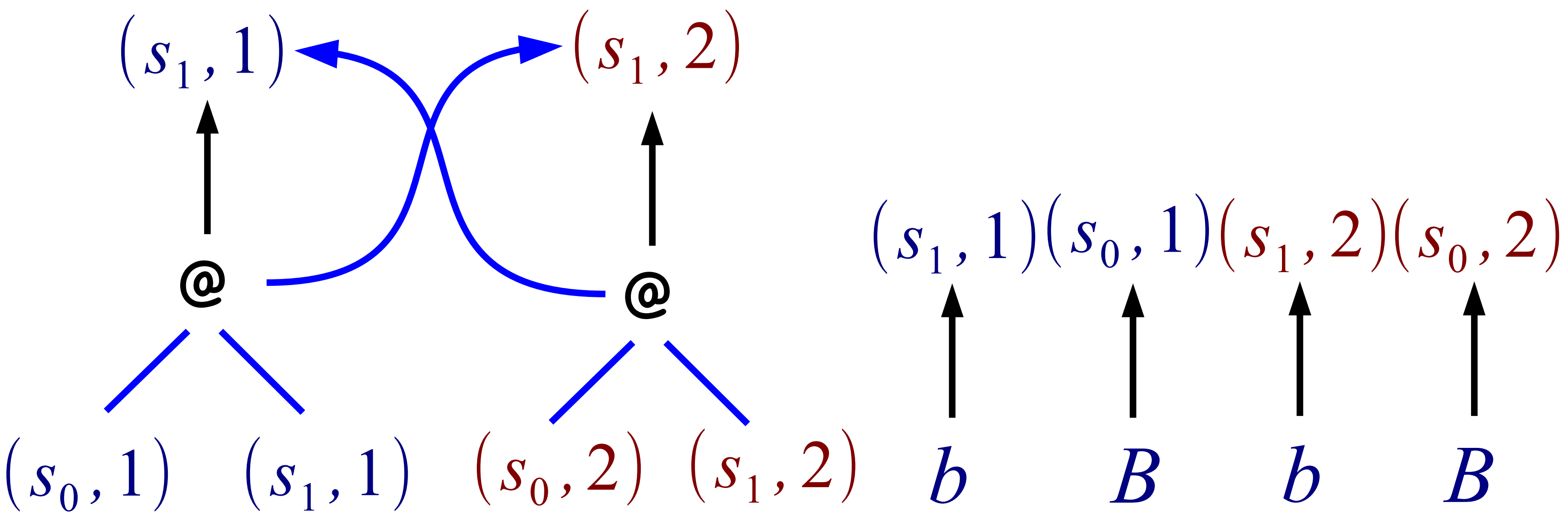
$$@\, s_0\, s_1 \rightarrow_B s_1 \quad b \rightarrow_B s_1 \quad B \rightarrow_B s_0$$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

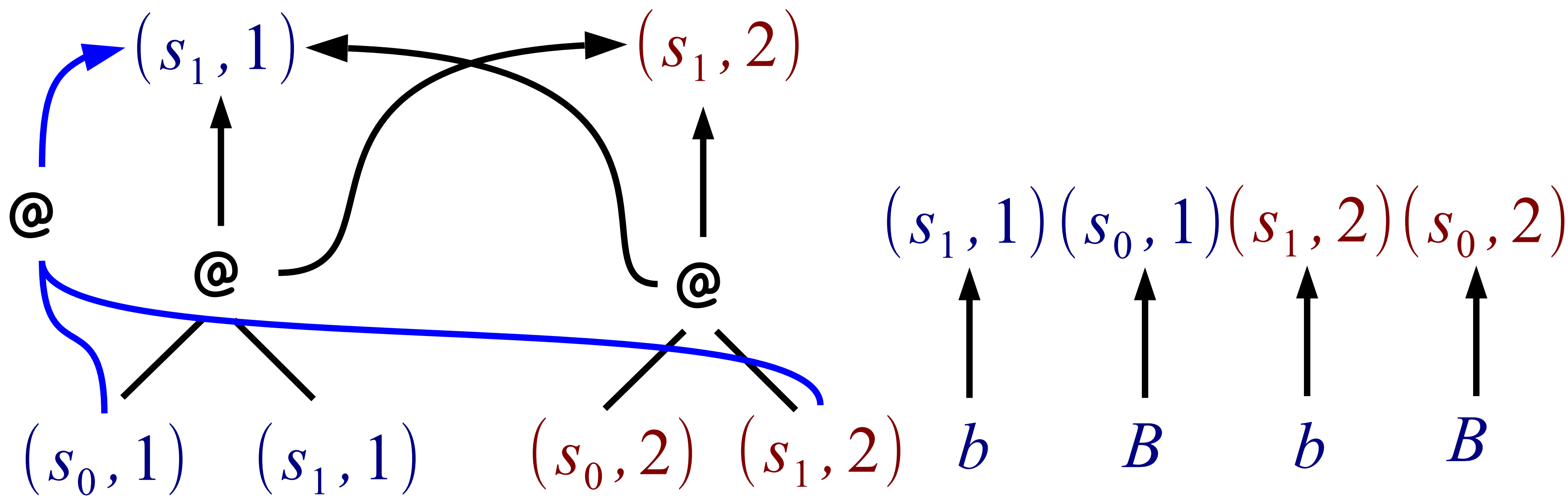
$$@\ s_0\ s_1 \rightarrow_B s_1 \quad b \rightarrow_B s_1 \quad B \rightarrow_B s_0$$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

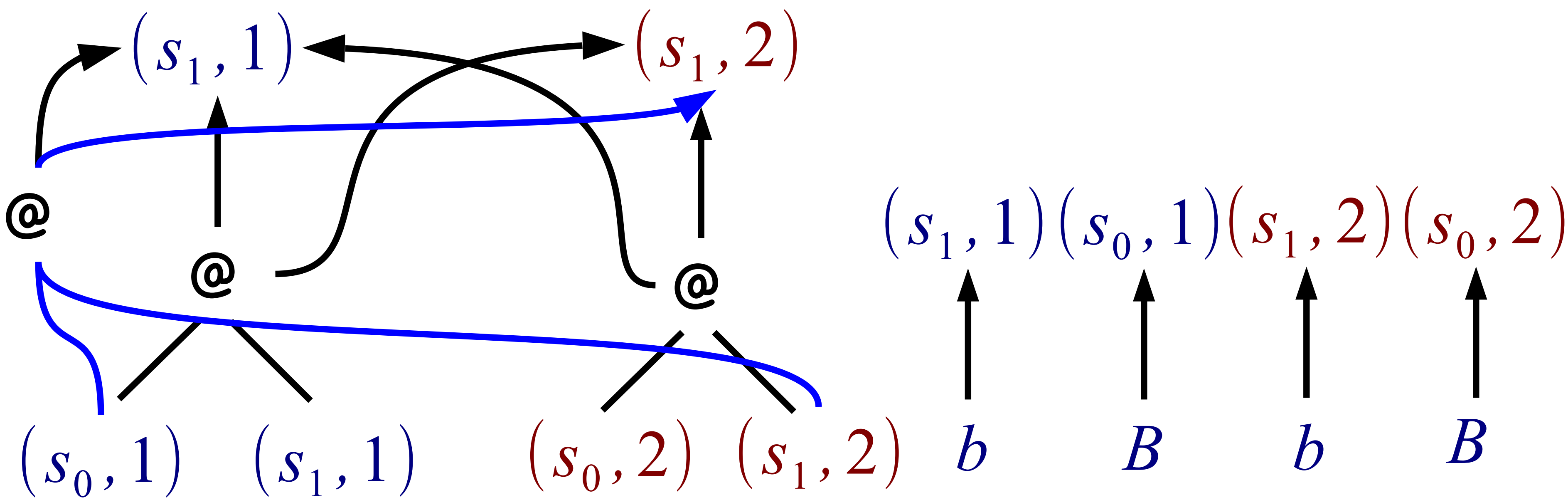
$$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

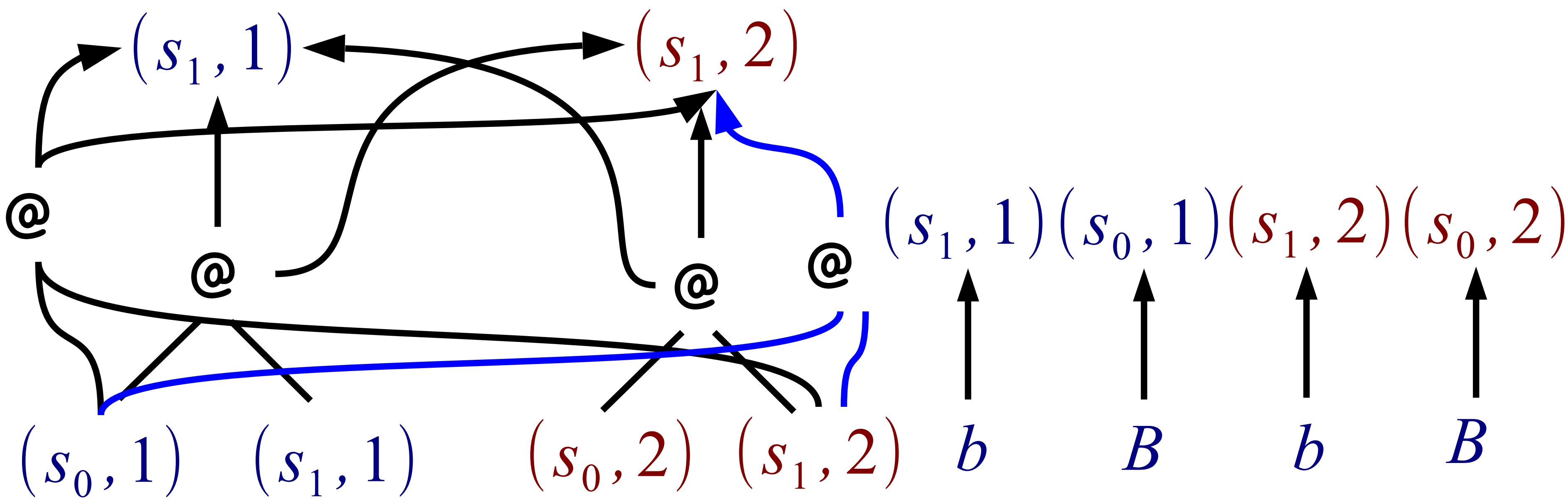
$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

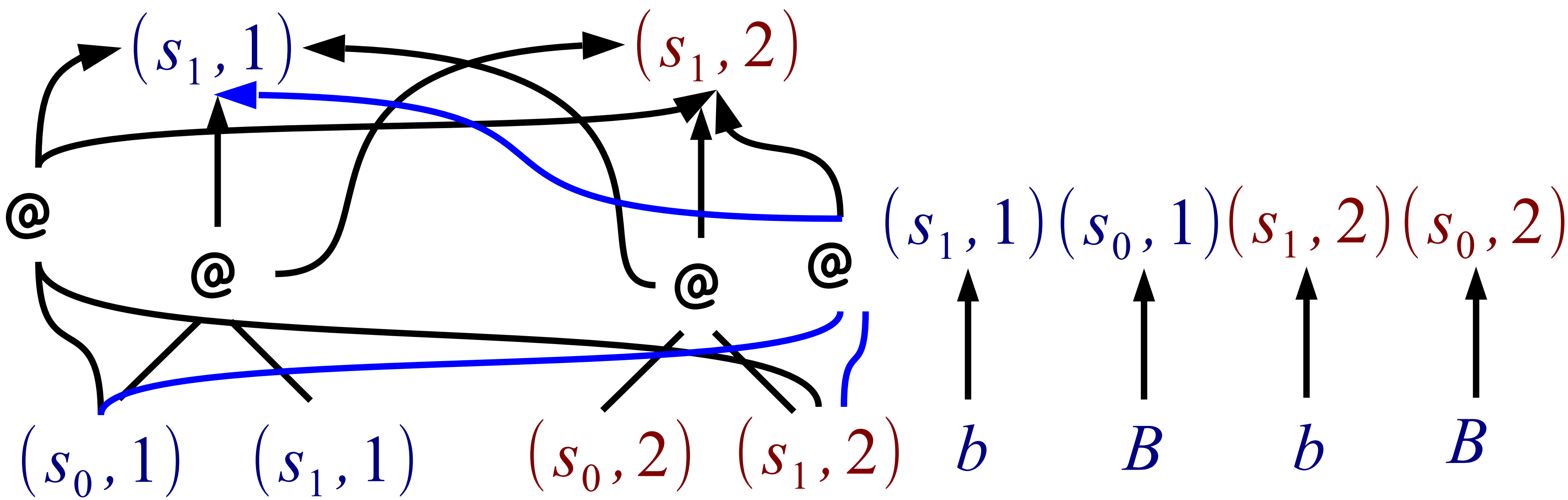
$$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

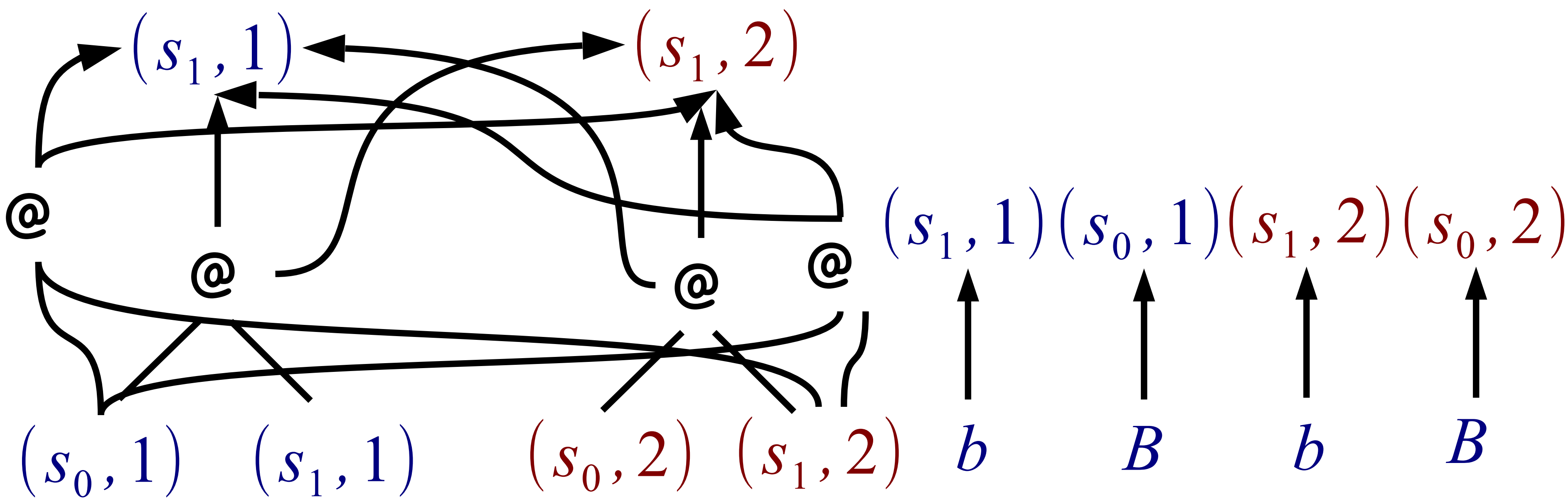
$$@\, s_0\, s_1 \rightarrow_B s_1 \quad b \rightarrow_B s_1 \quad B \rightarrow_B s_0$$



Cloning (2-copies) states and transitions of B

Example: Abstraction Refinement

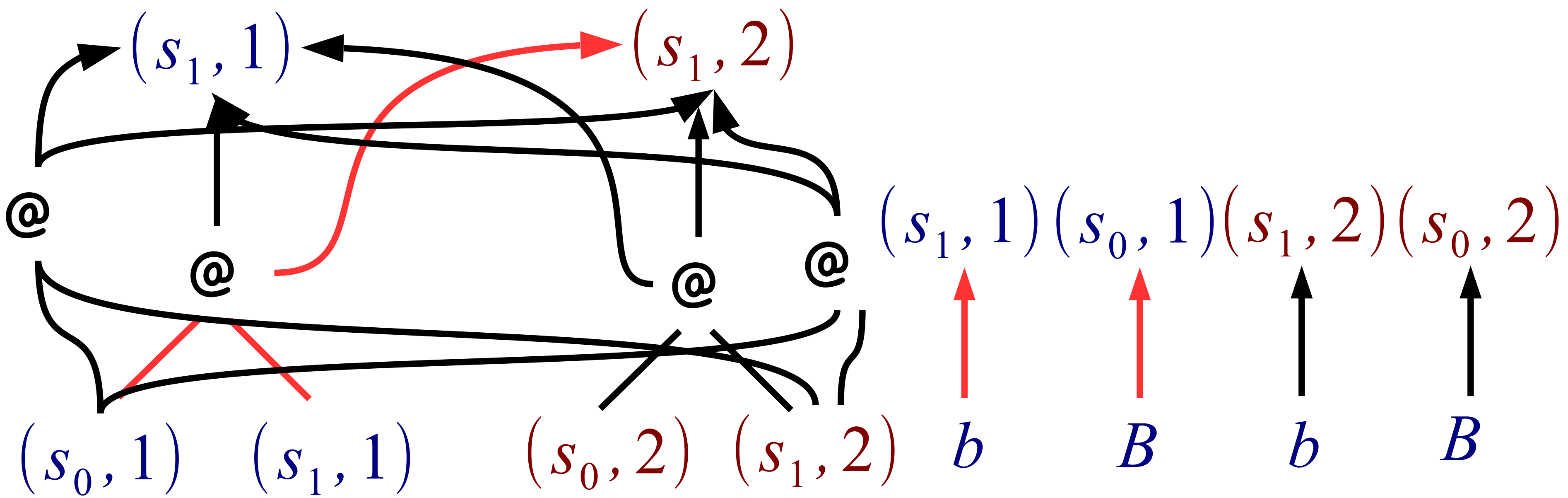
$$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$$



Those transitions are candidates for the refined automaton B'

Example: Abstraction Refinement

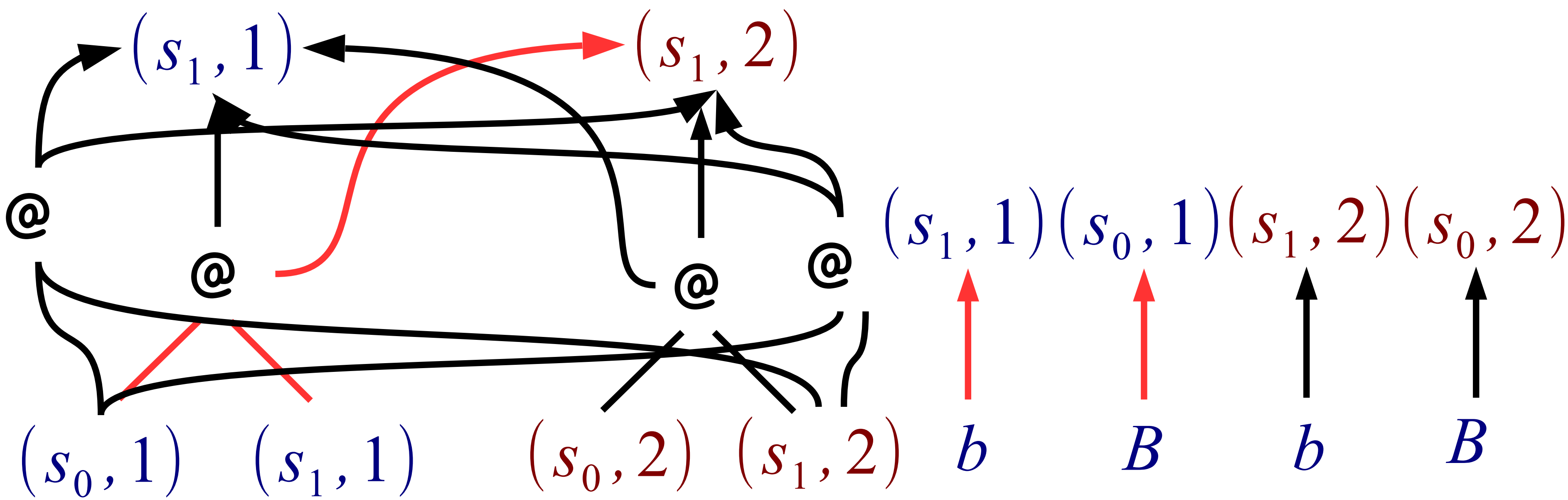
$$@\ s_0\ s_1 \to_B s_1 \quad b \to_B s_1 \quad B \to_B s_0$$



$$@(s_0,1)(s_1,1) \to_{B'} (s_1,2) \quad b \to_{B'} (s_1,1) \quad B \to_{B'} (s_0,1)$$

Example: Abstraction Refinement

$$@(s_0,1)(s_1,1) \rightarrow_{B'} (s_1,2) \quad b \rightarrow_{B'} (s_1,1) \quad B \rightarrow_{B'} (s_0,1)$$



The constraint is satisfied i.e., $\delta_{B'}(@Bb) \neq \delta_{B'}(b)$

How to Refine Abstraction

1. Cloning (k-copies) states and transitions of B
2. Extract a refined B' satisfying the constraint
3. If failed, increase k to k+1 and go to 1

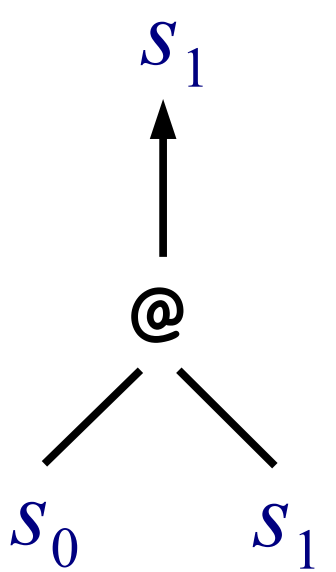
↓

Reduced to SMT Solving for formulas of uninterpreted functions

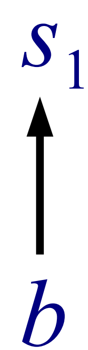
Example: Refinement by SMT Solving

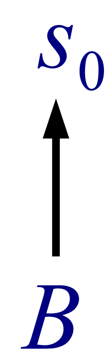
$$b \rightarrow_B s_1 \quad B \rightarrow_B s_0 \quad @\, s_0\, s_1 \rightarrow_B s_1$$

$$f_{s_0,@,s_1 s_2} : \{1,\ldots,k\}^2 \rightarrow \{1,\ldots,k\}$$



$$f_{s_1,b} : \{1,\ldots,k\} \qquad f_{s_0,B} : \{1,\ldots,k\}$$

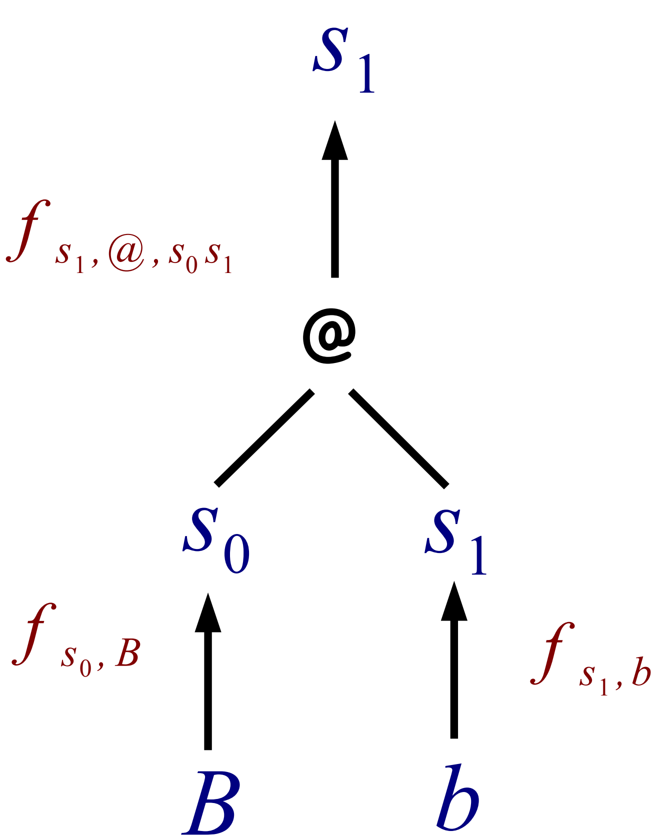




Generate an uninterpreted function on a finite domain {1,...,k} for each transition of B

Example: Refinement by SMT Solving

$$b \to_B s_1 \quad B \to_B s_0 \quad @\, s_0\, s_1 \to_B s_1$$



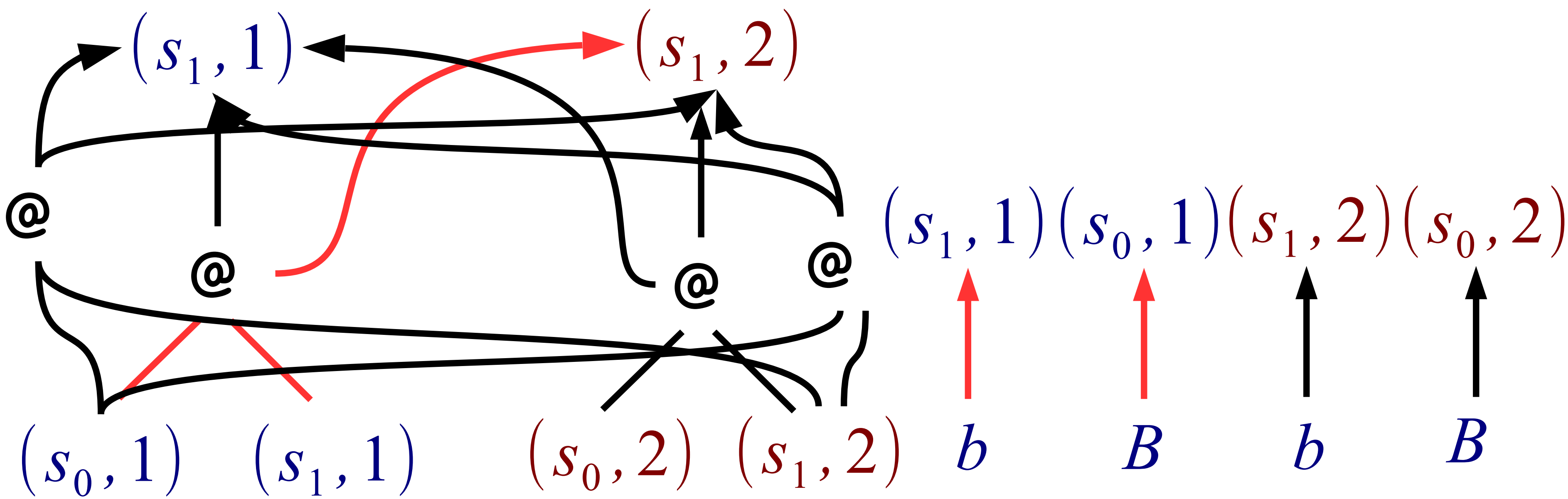
Refine B such that $\delta_B(@\, Bb) \neq \delta_B(b)$

SMT solving for the fomula of

$$f_{s_1,@,s_0 s_1}(f_{s_0,B}, f_{s_1,b}) \neq f_{s_1,b}$$

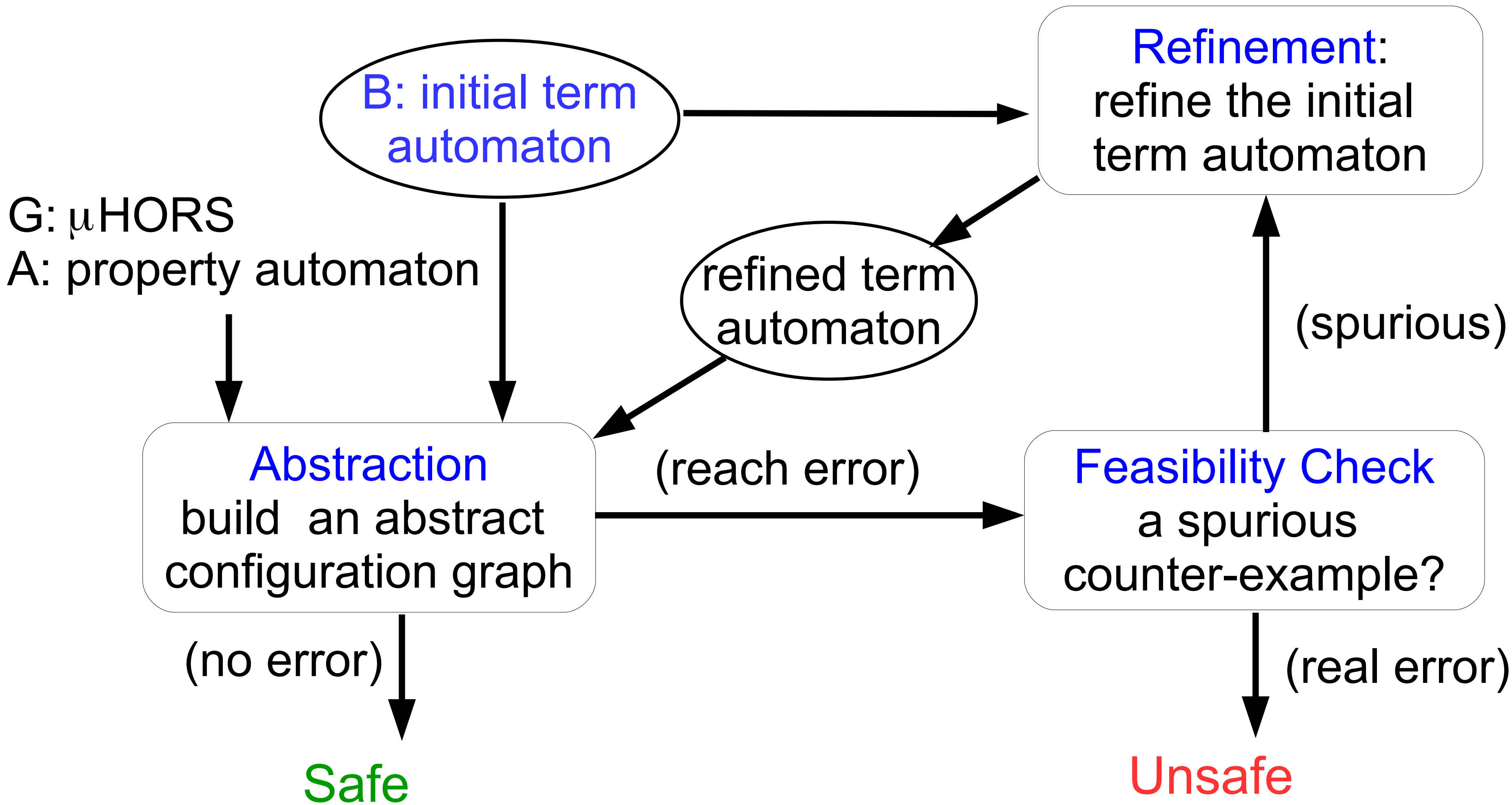
Example: Refinement by SMT Solving

$$f_{s_1,b}=1 \quad f_{s_0,B}=1 \quad f_{s_0,@,s_1 s_2}(1,1)=2 \quad f_{s_0,@,s_1 s_2}(i,j)=1\,(i\neq 1 \vee j\neq 1)$$



$$b \rightarrow_{B'} (s_1, f_{s_1,b}) \quad B \rightarrow_{B'} (s_0, f_{s_0,B}) \quad @(s_0,i)(s_1,j) \rightarrow_{B'} (s_1, f_{s_0,@,s_1 s_2}(i,j))$$

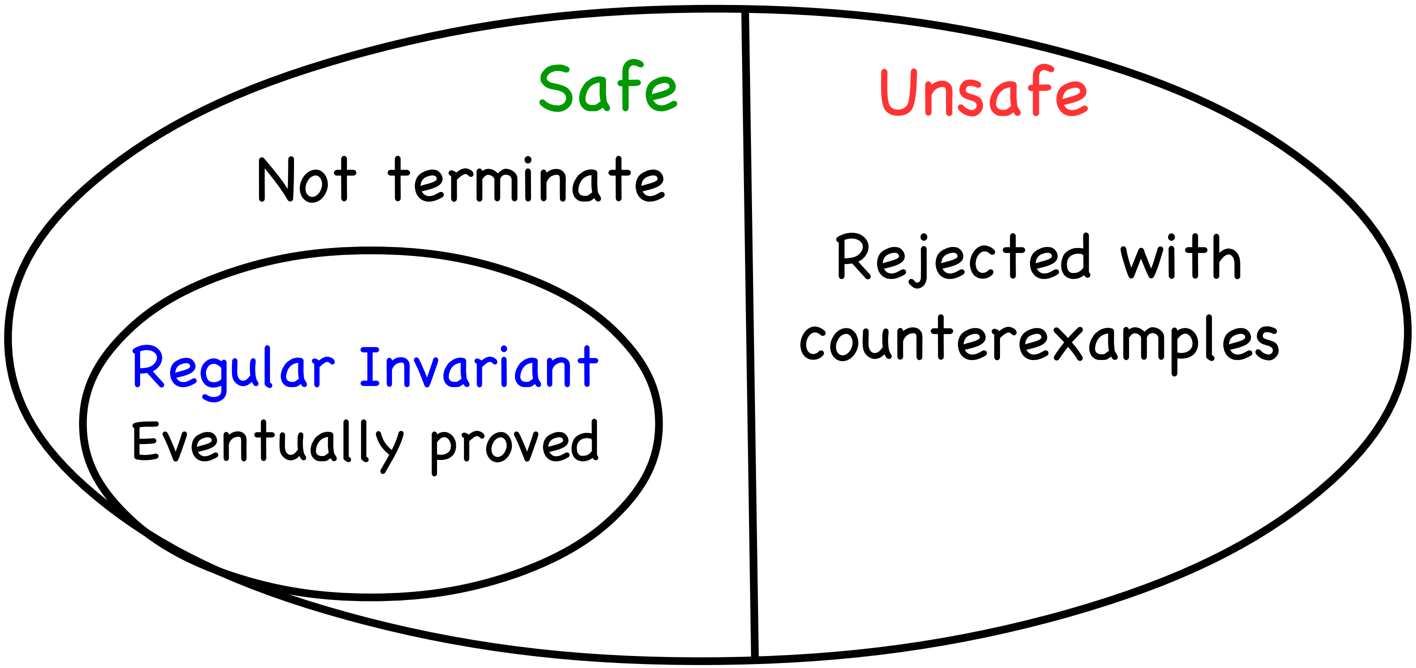
Our Approach: Summary of Key Ideas



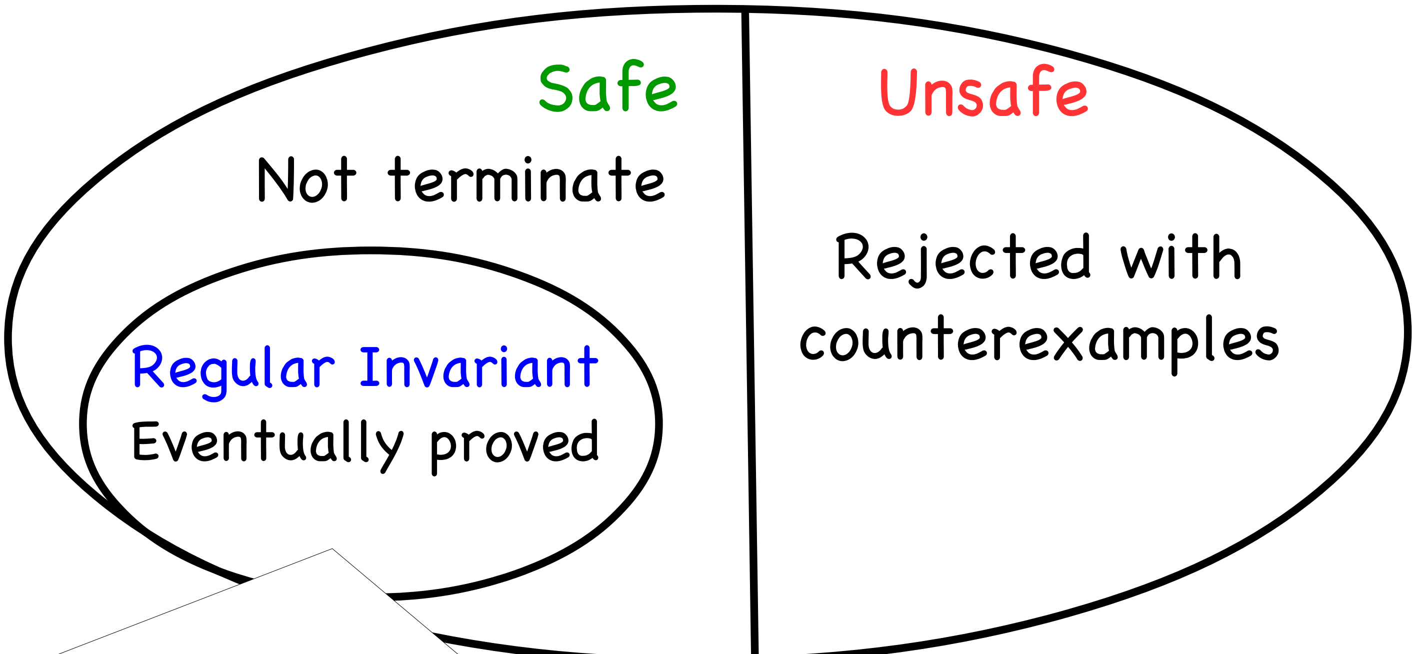
Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Example for abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Conclusion

Properties of Our Procedure



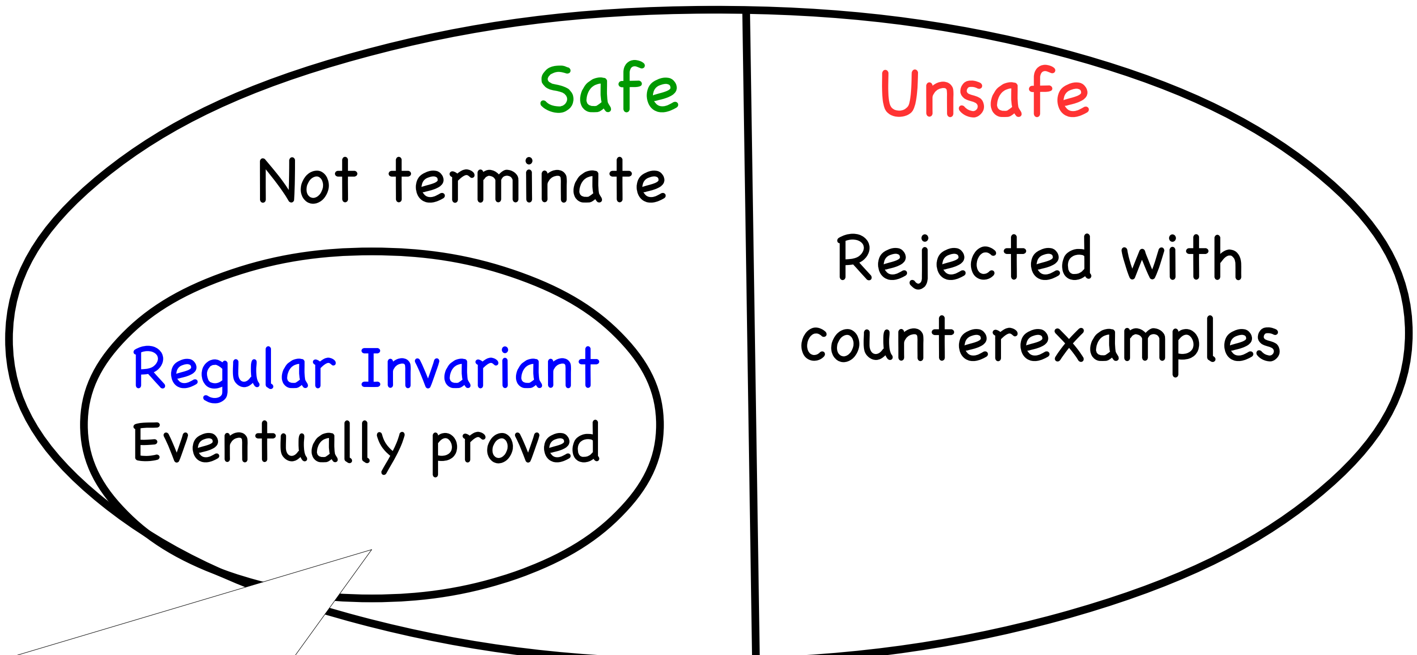
Properties of Our Procedure



Regular invariant I is a regular set of term trees satisfying:

1. $S \in I$
2. $t' \in I$ *if* $\exists t. t \rightarrow_G t' \wedge t \in I$
3. I contains no invalid term trees

Properties of Our Procedure

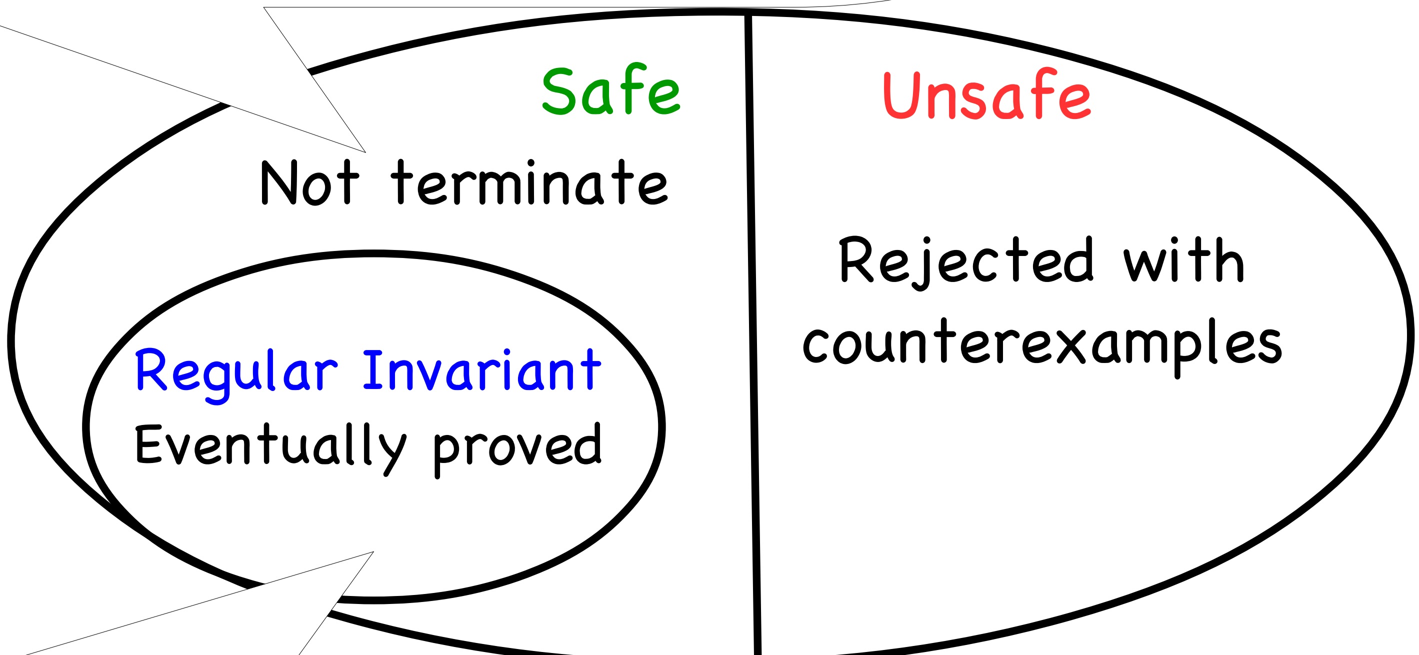


Relatively Complete:

A program P is proved safe <=>
There exists a regular invariant I

Incompleteness:
There exists a safe program with no regular invariants

…Procedure



Relatively Complete:
A program P is proved safe <=>
There exists a regular invariant I

Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Example for abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and conclusion

Implementation

- MuHorSar: model checker for μHORS by the automaton-based abstraction refinement
  - Z3 for SMT solving of uninterpreted functions
- A translator from multi-threaded boolean programs with recursion to μHORS
- Examine ways of building the initial term automaton by Sorts and Horsat
  - HorSat (backward saturation-based procedure for simply-typed HORS model checking [B., K., CSL13])

* Reuse the translator from Featherweight Java to μHORS [K.,I., ESOP13]

Example
Term Automata Built by Sorts

$$S \rightarrow F\,F\,b \qquad B\,h\,x \rightarrow b\,(h\,x)$$
$$F\,f\,g \rightarrow a\,(g\,(g\,c))\,(f\,f\,(B\,g))$$
$$S : o\,,\, B : (o \rightarrow o) \rightarrow o \rightarrow o$$
$$F : \mu\alpha.\,\alpha \rightarrow (o \rightarrow o) \rightarrow o$$

If ζ_o is the initial state, it accepts terms of sort o

$$a \rightarrow_B \zeta_{o \rightarrow o \rightarrow o}$$
$$b \rightarrow_B \zeta_{o \rightarrow o} \qquad c \rightarrow_B \zeta_o$$
$$B \rightarrow_B \zeta_{(o \rightarrow o) \rightarrow o \rightarrow o}$$
$$F \rightarrow_B \zeta_\kappa \qquad S \rightarrow_B \zeta_o$$
$$\textit{where } \kappa = \mu\alpha.\,\alpha \rightarrow (o \rightarrow o) \rightarrow o$$

$$@\,\zeta_\kappa\,\zeta_\kappa \rightarrow_B \zeta_{(o \rightarrow o) \rightarrow o}$$
$$@\,\zeta_{o \rightarrow o \rightarrow o}\,\zeta_o \rightarrow_B \zeta_{o \rightarrow o}$$
$$@\,\zeta_{(o \rightarrow o) \rightarrow o}\,\zeta_{o \rightarrow o} \rightarrow_B \zeta_o$$
$$@\,\zeta_{o \rightarrow o}\,\zeta_o \rightarrow_B \zeta_o$$
$$@\,\zeta_{(o \rightarrow o) \rightarrow o \rightarrow o}\,\zeta_{o \rightarrow o} \rightarrow_B \zeta_{o \rightarrow o}$$

Implementation

1. Run HorSat up to a bounded step of saturation
2. Get a type environment that finitely represents a subset of terms reducible to invalid terms
3. Build term automata from the type environment

- Building the initial term aut sorts (please see our paper) and Horsat
  - HorSat (backward saturation-based procedure for simply-typed HORS model checking [B., K., CSL13])

* Reuse the translator from Featherweight Java to μHORS [K.,I., ESOP13]

Experimental Results

for verifying Benchmarks from [K.,I., ESOP13]

Bench	#G	#A	R	MuHorSar (#Ar)	RTRecS
$\mathcal{G}_1$	2	2	Y	0.006	0.009
$\mathcal{G}_2$	3	2	Y	0.004	0.010
Thread	9	5	Y	0.013	0.181
Pred	15	1	Y	0.008	0.010
Ski1	22	1	N	0.008	0.005
Ski2	25	1	Y	0.008	0.010
L-append	30	1	Y	0.013	0.012
L-map	182	1	Y	0.561	0.189
L-app-map	212	1	Y	0.840	0.279
L-even	87	1	Y	0.077	0.021
L-filter	122	1	Y	1.454 (6)	0.429
L-risers	122	1	Y	1.450 (6)	0.431
Twofiles	21	5	Y	0.027	4.390

- #G: number of rules, #A: state size of tree automaton, #R: result
- Time excludes the translation from Featherweight Java to uHORS
- Mac OS X v.10.9.2, 1.7 GHz Intel Core i7 processor, 8GB RAM

Experimental Results

for verifying new Featherweight Java programs

Bench	#G	#A	R	MuHorSar (#Ar)	RTRecS
stack	33	1	Y	0.040	0.207
		3		0.039	3.435
		5		0.044	23.292
stack-br	39	1	Y	0.396 (13)	-
		3		0.403 (13)	-
		5		0.397 (13)	-
queue	56	1	Y	0.169	0.143
		3		0.173	2.140
		5		0.164	12.633
queue-br	61	1	Y	0.249 (2)	-
		3		0.249 (2)	-
		5		0.249 (2)	-
queue-pc	104	1	Y	1.160	0.211
		3		1.218	0.642
		5		1.137	1.648
2stack-e	52	1	N	0.105	-
		5		0.105	-
2stack-pc	88	1	Y	4.202	0.498
		5		4.176	1.262
		7		4.182	2.132
nat	35	1	Y	17.810 (147)	0.288

"-": time out for 5 mins

* MuHorSar scales well as the size of the property automaton increases, but RTRecs does not

Experimental Results

for Multi-threaded boolean programs with recursion

Bench	#G	#A	R	MuHorSar (#Ar)	RTRecS
locks-e	103	5	N	0.160	-
dining-e	135	5	N	2.857 (28)	-
dining-sp-e	193	5	N	10.997 (90)	-
bluetooth	129	1	N	2.300 (25)	-
bluetooth-v	158	1	N	272.626 (326)	-
locks	95	5	Y	0.779	-
plotter	88	4	Y	0.195	1.189
peterson	74	2	Y	3.331 (2)	-
peterson-d	80	9	Y	-	-
dekker	94	2	Y	-	-
pc-monitor	71	5	Y	0.338	-
pc-sp	111	5	Y	2.250	-
dining-sp	303	5	Y	-	-

* MuHorSar is effective in counterexample finding but RTRecs does not

Experimental Results

for Multi-threaded boolean programs with recursion

Bench	#G	#A	R	MUHORSAR (#Ar)	
locks-e	103	5	N	0.160	0.161
dining-e	135	5	N	2.857 (28)	0.582
dining-sp-e	193	5	N	10.997 (90)	0.961
bluetooth	129	1	N	2.300 (25)	1.693 (9)
bluetooth-v	158	1	N	272.626 (326)	4.223 (20)
locks	95	5	Y	0.779	0.247
plotter	88	4	Y	0.195	0.251
peterson	74	2	Y	3.331 (2)	0.467
peterson-d	80	9	Y	-	6.971 (5)
dekker	94	2	Y	-	0.473
pc-monitor	71	5	Y	0.338	0.217
pc-sp	111	5	Y	2.250	0.207
dining-sp	303	5	Y	-	18.963

* MuHorSar is effective in counterexample finding but RTRecs does not

Outline

- Background
  - μHORS model checking
  - Example: application to OO verification
- New model checking procedure for μHORS
  - Overview and key ideas
  - Example for abstraction and refinement
  - Properties of the procedure
- Implementation and experiments
- Related work and conclusion

Related Work

- Inspired by two state-of-the-art procedures for ordinary (simply-typed) HORS
  - Preface: type-directed abstraction refinement [Ramsey+, POPL14]
  - HorSat: backward saturation-based procedure (not terminate for μHORS) [Broadbent, Kobayashi, CSL13]
- Tree automata completion [Jacquemard, RTA96]
  - Reachability analysis of term rewriting systems
  - Applicable to μHORS model checking, but no discussion on relative completeness condition

Conclusion

- A new model checking procedure for μHORS based on automata-based abstraction refinement
  - ✓ Sound & relatively-complete w.r.t. regular invariants
  - ✓ Often scales better than RTRecs [K., I.,ESOP13]
- Relative completeness by regular invariants is equivalent to that by typability (in a recursive intersection type system)
- Application to OO and multi-threaded boolean programs with recursion