Verifying Relational Properties of Functional Programs by First-Order Refinement

Ryosuke Sato

Joint work with **Kazuyuki Asada** and **Naoki Kobayashi**

This Talk

- **Automated** verification of **relational properties** for higher-order programs
  - Example:

```
let rec sum n =
    if n = 0 then 0 else n + sum (n-1)
let rec sumacc a n =
    if n = 0 then a else sumacc (n+a) (n-1)
```

$\vdash^{?} (\mathrm{sum}, \mathrm{sumacc}\ 0) : \{(f, g) : (\mathrm{int} \to \mathrm{int})^2 \mid \forall n.\, f(n) = \mathrm{g}(n)\}$

$\tau^2 = \tau \times \tau$

Outline

- Motivation and approach
- Overview (through examples)
- More details
- Experiments
- Related work
- Conclusion

Relational Properties

- **Relationship among multiple functions**
  - **e.g., function equality:**
    $\{(f, g) : (\text{int} \to \text{int})^2 \mid \forall n.\, f(n) = g(n)\}$
- **Relationship among multiple invocations of a function**
  - **e.g., monotonicity:**
    $\{f : \text{int} \to \text{int} \mid \forall n.\, f(n) \leq f(n+1)\}$

Existing Automated Verifiers

- Verifiers based on first-order refinement types [Rondon+ '08; K+ '11; Zhu & Jagannathan '13]
  - Refinement type: type refined by predicates $\{x\colon \sigma \mid P\}$
    - First-order: no functions & no quantifiers in P
  - Relational properties cannot be represented as first-order refinement types
- Verifier for first-order functional programs [Suter+ '11]
  - Can deal with some relational properties
  - Cannot deal with higher-order programs

Our Approach

- Reduce **general** refinement type checking

$$\vdash^{?} t : \tau$$

  to **first-order** refinement type checking

$$\vdash^{?} (t)^{\#} : (\tau)^{\#}$$

- Solve $\vdash^{?} (t)^{\#} : (\tau)^{\#}$ by an existing verifier (e.g. MoCHi [K+ '11; S+ '13])

Outline

- Motivation and approach
- **Overview (through examples)**
- **More details**
- **Experiments**
- **Related work**
- **Conclusion**

Example 1

```
let rec sum n =
    if n = 0 then 0 else n + sum (n-1)
let rec sumacc a n =
    if n = 0 then a else sumacc (n+a) (n-1)
```

$$\vdash^{?} (\mathrm{sum}, \mathrm{sumacc}\ \mathbf{0}):$$
$$\{(f, g): (\mathrm{int} \to \mathrm{int})^2 \mid \forall n.\, f(n) = g(n)\}$$

Remove ∀ and Function Symbols

$$\vdash^{?} (\mathbf{sum}, \mathbf{sumacc}\ \mathbf{0}):$$

$$\left\{(f, g): (\mathbf{int} \to \mathbf{int})^2 \;\middle|\; \forall n.\, f(n) = g(n)\right\}$$

$$\vdash^{?} \lambda \mathbf{n}.\, (\mathbf{sum}\ n, \mathbf{sumacc}\ \mathbf{0}\ n):$$

$$(n: \mathbf{int}) \to \left\{(r, s): \mathbf{int}^2 \;\middle|\; r = s\right\}$$

Remove ∀ and Function Symbols

$\vdash^{?} (\mathbf{sum}, \mathbf{sumacc}\ \mathbf{0})$:

$$\{(f, g): (\mathbf{int} \to \mathbf{int})^2 \mid \forall n.\ f(n) = g(n)\}$$

$\vdash^{?} \lambda \mathbf{n}.\ (\mathbf{sum}\ n, \mathbf{sumacc}\ \mathbf{0}\ n)$:

$$(n: \mathbf{int}) \to \{(r, s): \mathbf{int}^2 \mid r = s\}$$

First-order refinement

Difficulty of verifying relational properties in first-order refinement

To verify

$$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n) : (n\colon \mathrm{int}) \to \left\{(r, s)\colon \mathrm{int}^2 \;\middle|\; r = s\right\},$$

we have to infer

$$\mathrm{sum} : \{n\colon \mathrm{int} \mid P[n]\} \to \{r\colon \mathrm{int} \mid R[n, r]\}$$

$$\mathrm{sumacc}\ 0 : \{n\colon \mathrm{int} \mid Q[n]\} \to \{s\colon \mathrm{int} \mid S[n, s]\}$$

such that $R[n, r] \wedge S[n, s] \Rightarrow r = s$

$$\frac{\vdots \qquad \vdots}{\Gamma \vdash \mathrm{sum}\ n : \{r\colon \mathrm{int} \mid R[n, r]\} \quad \Gamma \vdash \mathrm{sumacc}\ 0\ n : \{s\colon \mathrm{int} \mid S[n, s]\}}{\Gamma \vdash (\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n) : \left\{(r, s)\colon \mathrm{int}^2 \;\middle|\; r = s\right\}}$$

Difficulty of verifying relational properties in first-order refinement

To verify

$$\vdash^{?} \lambda \mathrm{n}.\ (\mathrm{sum}\ n, \mathrm{su}$$
$$(n\colon \mathrm{int}) \to \{$$

$r = \dfrac{n(n+1)}{2}$

$s = \dfrac{n(n+1)}{2}$

we have to infer

$$\mathrm{sum} : \{n\colon \mathrm{int} \mid P[n]\} \to \{r\colon \mathrm{int} \mid R[n, r]\}$$
$$\mathrm{sumacc}\ 0 : \{n\colon \mathrm{int} \mid Q[n]\} \to \{s\colon \mathrm{int} \mid S[n, s]\}$$

such that $R[n, r] \wedge S[n, s] \Rightarrow r = s$

$$\frac{\dfrac{\vdots}{\Gamma \vdash \mathrm{sum}\ n : \{r\colon \mathrm{int} \mid R[n, r]\}} \quad \dfrac{\vdots}{\Gamma \vdash \mathrm{sumacc}\ 0\ n : \{s\colon \mathrm{int} \mid S[n, s]\}}}{\Gamma \vdash (\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n) : \left\{(r, s)\colon \mathrm{int}^2 \;\middle|\; r = s\right\}}$$

Tupling sum & sumacc

$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n)$:

$$(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$$

Calc sum & sumacc simultaneously

```
let rec sum_sumacc a n =
    if n = 0 then (0, a) else
    let (r, s) = sum_sumacc (n+a) (n-1) in
    (n+r, s)
```

$\vdash^{?}$ sum_sumacc 0 : $(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$

Tupling sum & sumacc

$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n)$:

$$(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$$

Calc sum & sumacc simultaneously

```
let rec sum_sumacc a n =
    if n = 0 then (0, a) else
    let (r, s) = sum_sumacc (n+a) (n-1) in
    (n+r, s)
```

$\vdash^{?}$ sum_sumacc 0 : $(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$

Tupling sum & sumacc

$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n)$:

$$(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$$

Calc sum & sumacc simultaneously

```
let rec sum_sumacc a n =
    if n = 0 then (0, a) else
    let (r, s) = sum_sumacc (n+a) (n-1) in
    (n+r, s)
```

$\vdash^{?}$ sum_sumacc 0 : $(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$

Tupling sum & sumacc

$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n)\colon$

$$(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$$

Calc sum & sumacc simultaneously

```
let rec sum_sumacc a n =
    if n = 0 then (0, a) else
    let (r, s) = sum_sumacc (n+a) (n-1) in
    (n+r, s)
```

$\vdash^{?}$ sum_sumacc 0 : $(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$

Tupling sum & sumacc

$\vdash^{?} \lambda \mathrm{n}.\,(\mathrm{sum}\ n, \mathrm{sumacc}\ 0\ n)$:

$$(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$$

```
let rec sum_sumacc a n =
    if n = 0 then (0, a) else
    let (r, s) = sum_sumacc (n+a) (n-1) in
    (n+r, s)
```

Calc sum & sumacc simultaneously

$\vdash^{?}$ sum_sumacc 0 : $(n\colon \mathrm{int}) \to \{(r, s)\colon \mathrm{int}^2 \mid r = s\}$

Can be verified by an existing verifier

Example 2

```
let rec append x y =
    match x(0) with ⊥ -> y
    | Some(n) -> cons n (append (tail x) y)
```

$\vdash^{?}$ append :

int$_\perp$ = int option

None

$$(x\colon \text{int} \to \text{int}_\perp) \to \{y\colon \text{int} \to \text{int}_\perp \mid y(0) = \perp\}$$
$$\to \{r\colon \text{int} \to \text{int}_\perp \mid \forall i.\, r(i) = x(i)\}$$

append x [] = x

Remove Dependencies between Arguments and Return Types

$$\vdash^{?} \mathrm{append} :$$

$$(x\colon \mathrm{int} \to \mathrm{int}_{\perp}) \to \{y\colon \mathrm{int} \to \mathrm{int}_{\perp} \mid y(0) = \perp\}$$
$$\to \{r\colon \mathrm{int} \to \mathrm{int}_{\perp} \mid \forall i.\, r(i) = x(i)\}$$

$$\vdash^{?} \lambda x, y.\, (x, \mathrm{append}\ x\ y) :$$

$$(x\colon \mathrm{int} \to \mathrm{int}_{\perp}) \to \{y\colon \mathrm{int} \to \mathrm{int}_{\perp} \mid y(0) = \perp\}$$
$$\to \left\{(x', r) : (\mathrm{int} \to \mathrm{int}_{\perp})^2 \;\middle|\; \forall i.\, r(i) = x'(i)\right\}$$

Remove Dependencies between Arguments and Return Types

$\vdash^{?}$ append :

$$(x\colon \mathrm{int} \to \mathrm{int}_\perp) \to \{y\colon \mathrm{int} \to \mathrm{int}_\perp \mid y(0) = \perp\} \to \{r\colon \mathrm{int} \to \mathrm{int}_\perp \mid \forall i.\, r(i) = x(i)\}$$

$\vdash^{?} \lambda x, y.\, (x,\ \mathrm{append}\ x\ y)$:

$$(x\colon \mathrm{int} \to \mathrm{int}_\perp) \to \{y\colon \mathrm{int} \to \mathrm{int}_\perp \mid y(0) = \perp\} \to \left\{(x', r) : (\mathrm{int} \to \mathrm{int}_\perp)^2 \;\middle|\; \forall i.\, r(i) = x'(i)\right\}$$

Cf. $\vdash^{?} (\mathrm{sum}, \mathrm{sumacc}\ 0)$:

$$\{(f, g)\colon (\mathrm{int} \to \mathrm{int})^2 \mid \forall n.\, f(n) = g(n)\}$$

Remove ∀ and Function Symbols

$$\vdash^{?} \lambda x, y.\,(x, \mathrm{append}\ x\ y) :$$
$$(x\colon \mathrm{int} \to \mathrm{int}_\perp) \to \{y\colon \mathrm{int} \to \mathrm{int}_\perp \mid y(0) = \perp\}$$
$$\to \{(x' \times r) : (\mathrm{int} \to \mathrm{int}_\perp)^2 \mid \forall i.\, r(i) = x'(i)\}$$

$$\vdash^{?} \lambda x, y.\,\lambda(i, j).\,(x(i), (\mathrm{append}\ x\ y)(j)) :$$
$$(x\colon \mathrm{int} \to \mathrm{int}_\perp) \to (y\colon (j\colon \mathrm{int}) \to \{u\colon \mathrm{int}_\perp \mid j = 0 \Rightarrow u = \perp\})$$
$$\to (i\colon \mathrm{int}) \to \{(r, s)\colon (\mathrm{int}_\perp)^2 \mid s = r\}$$

$$\vdash^{?} \lambda x, y.\ \mathit{Tupling}(x, \mathrm{append}\ x\ y) : \text{(the same type)}$$

Remove ∀ and Function Symbols

$$\vdash^{?} \lambda x, y.\,(x, \mathrm{append}\ x\ y) :$$
$$(x\colon \mathrm{int} \to \mathrm{int}_{\perp}) \to \{y\colon \mathrm{int} \to \mathrm{int}_{\perp} \mid y(0) = \perp\}$$
$$\to \{(x' \times r) : (\mathrm{int} \to \mathrm{int}_{\perp})^2 \mid \forall i.\, r(i) = x'(i)\}$$

$$\vdash^{?} \lambda x, y.\, \lambda(i, j).\,(x(i), (\mathrm{append}\ x\ y)(j)) :$$
$$(x\colon \mathrm{int} \to \mathrm{int}_{\perp}) \to (y\colon (j\colon \mathrm{int}) \to \{u\colon \mathrm{int}_{\perp} \mid j = 0 \Rightarrow u = \perp\})$$
$$\to (i\colon \mathrm{int}) \to \{(r, s)\colon (\mathrm{int}_{\perp})^2 \mid s = r\}$$

$$\vdash^{?} \lambda x, y.\, \mathit{Tupling}(x, \mathrm{append}\ x\ y) : \textbf{(the same type)}$$

Outline

- Motivation and approach
- Overview (through examples)
- **More details**
- **Experiments**
- **Related work**
- **Conclusion**

Two Kinds of Transformations

- Transformation for adjusting specification
  - Reduction from general to first-order refinement
- Auxiliary transformation to help automated verification
  - Tupling
  - Insertion of assume statements

Transformation for adjusting specification

- **Removal of dependencies between function arguments and return types:**

$\vdash^? \lambda f.t \quad : (f:\tau) \to \{r:\sigma \mid P[f,r]\}$

$\vdash^? \lambda f.(f,t) : (f:\tau) \to (f':\tau) \times \{r:\sigma \mid P[f',r]\}$

- **Linearization of function symbols:**

$\vdash^? t \quad : \{f:\mathrm{int} \to \mathrm{int} \mid P[f(x), f(y)]\}$

$\vdash^? (t,t) : \{(f,g):(\mathrm{int} \to \mathrm{int})^2 \mid P[f(x), g(y)]\}$

- **Elimination of $\forall$ and functions:**

$\vdash^? (\lambda x.t_1, \lambda y.t_2) : \{(f,g):(\mathrm{int} \to \mathrm{int})^2 \mid \forall x,y.\, P[f(x), g(y)]\}$

$\vdash^? \lambda(x,y).(t_1,t_2) : ((x,y):\mathrm{int}^2) \to \{(r,s):\mathrm{int}^2 \mid P[r,s]\}$

Assume linearity

Transformation for adjusting specification

$\vdash^? \mathbf{sum} : \{f : \mathrm{int} \to \mathrm{int} \mid f(x) = x + f(x-1)\}$

$\vdash^? (\mathbf{sum}, \mathrm{sum}) :$
$\{(f, g) : (\mathrm{int} \to \mathrm{int})^2 \mid f(x) = x + g(x-1)\}$

- **Linearization of function symbols:**

$\vdash^? t : \{f : \mathrm{int} \to \mathrm{int} \mid P[f(x), f(y)]\}$
$\vdash^? (t, t) : \{(f, g) : (\mathrm{int} \to \mathrm{int})^2 \mid P[f(x), g(y)]\}$

Assume linearity

- **Elimination of ∀ and functions:**

$\vdash^? (\lambda x.\, t_1, \lambda y.\, t_2) : \{(f, g) : (\mathrm{int} \to \mathrm{int})^2 \mid \forall x, y.\, P[f(x), g(y)]\}$
$\vdash^? \lambda(x, y).\, (t_1, t_2) : ((x, y) : \mathrm{int}^2) \to \{(r, s) : \mathrm{int}^2 \mid P[r, s]\}$

Two Kinds of Transformations

- Transformation for adjusting specification
  - Reduction from general to first-order refinement
- **Auxiliary transformation to help automated verification**
  - **Tupling**
  - **Insertion of assume statements**

Auxiliary transformation to help automated verification

- Tupling of recursive functions

  $\lambda(x, y).(f\ x, g\ y)$ ⇒ fg

- Insertion of assume statements
  - About the property of tupled functions
    - Deterministic
    - fg (x,y) = (f x, g y)

Example:
Insertion of assume statements

$$\vdash^{?} h : \{(f,g) : (\text{int} \to \text{int})^2 \mid \forall x, y.\, x \le y \Rightarrow f(x) \le g(y)\} \to \{d : \text{int} \mid d \ge 0\}$$

```
let h (f,g) = ... let r = f x in ...
  if x < y then let s = g y in s - r else ...
```

$$\vdash^{?} h : \left((x,y) : \text{int}^2 \to \{(r,s) : \text{int}^2 \mid x \le y \Rightarrow r \le s\}\right) \to \{d : \text{int} \mid d \ge 0\}$$

dummy value

```
let h fg = ... let (r1, s1) = fg (x, ⊥) in ...
  if x < y then let (r2, s2) = fg (x, y) in
    assume (r1 = r2); s2 - r1 else ...
```

✗ fg (⊥, y)

Example: Insertion of assume statements

When checking $\vdash^{?} s2 - r1 : \{d : \text{int} \mid d \geq 0\}$, a verifier knows that

- $x < y$
- $r2 \leq s2$

$\vdash^{?} h : \left((x, y) : \text{int}^2 \to \{(r, s) : \text{int}^2 \mid x \leq y \Rightarrow r \leq s\}\right)$
$\to \{d : \text{int} \mid d \geq 0\}$

dummy value

```
let h fg = ... let (r1, s1) = fg (x, ⊥) in ...
  if x < y then let (r2, s2) = fg (x, y) in
    assume (r1 = r2); s2 - r1 else ...
```

✗ fg (⊥, y)

Example:
Insertion of assume statements

When checking $\vdash^{?} s2 - r1 : \{d: \text{int} \mid d \geq 0\}$, $y)\}$
a verifier knows that

- $x < y$
- $r2 \leq s2$
- $r1 = r2$

$\Rightarrow r1 \leq s2 \Rightarrow s2 - r1 \geq 0$

$$\vdash^{?} h : \left((x, y): \text{int}^2 \rightarrow \left\{(r, s): \text{int}^2 \middle| x \leq y \Rightarrow r \leq s\right\}\right)$$
$$\rightarrow \{d: \text{int} \mid d \geq 0\}$$

dummy value

```
let h fg = ... let (r1, s1) = fg (x, ⊥) in ...
  if x < y then let (r2, s2) = fg (x, y) in
    assume (r1 = r2); s2 - r1 else ...
```

✗ fg (⊥, y)

Soundness of Verification by Transformation

Theorem:

Suppose τ is at most order-2.

Then,

$$\vDash (t)^{\#} : (\tau)^{\#} \quad \Rightarrow \quad \vDash t : \tau \ .$$

Note: Functions of arbitrary order can be used inside t

Outline

- Motivation and approach
- Overview (through examples)
- More details
- **Experiments**
- **Related work**
- **Conclusion**

Experiments

program	size (before #) [words]	size (after #) [words]	predicate	time [sec]
sum-acc	56	282	0	0.54
sum-simple	40	270	0	0.75
sum-mono	27	279	0	0.45
mult-acc	63	347	0	0.38
a-max-gen	112	476	1	0.29
append-xs-nil	72	1364	0	45.57
append-nil-xs	63	725	0	16.43
rev	128	1868	0	176.24
insert	32	6262	0	52.49
insertsort	38	7044	2	14.33

- Implemented as an extension of MoCHi [K+ '11, S+ '13]
- Intel Core i7-3930 CPU, 16 GB memory

Limitations

- Due to the underlying verifier MoCHi
  - The current implementation of MoCHi cannot infer appropriate predicates for some programs (e.g., merge sort)
- Due to our transformation
  - $\{v : \sigma \mid \exists x. P\}$

Related Work

- (Semi-)automated refinement type checking that supports uninterpreted functions [Rondon+ '08; Zhu & Jagannathan '13]
  - Cannot deal with relational properties
- Verification of first-order functional programs [Suter+ '11]
  - Can deal with some relational properties
  - Cannot deal with higher-order programs

Conclusion

- **Automated** verification of **relational properties**
  - **General refinement types** for representing relational properties
  - **Reduction** to **first-order refinement** type checking
  - **Tupling** for enabling first-order refinement type checking