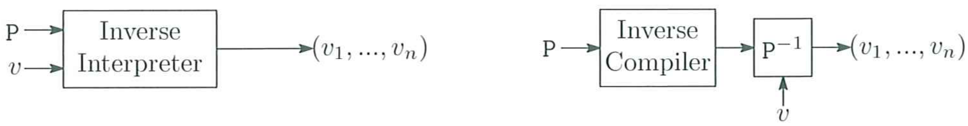
# Program Generation Techniques for Developing Reliable Software

Sakabe Laboratory & Sakai Laboratory

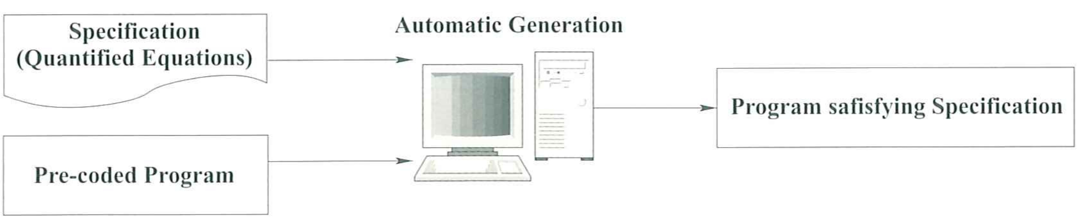
Graduate School of Information Science, Nagoya University
{sakabe,sakai}@is.nagoya-u.ac.jp

Programs are usually developed from given specifications having sufficient information for their coding. Manual coding is unavoidable for obtaining actually runnable programs. To decrease this tiresome work, we are investigating program generation methods. In this presentation, we introduce two generation methods: The first method is an inverse compiler that automatically generates an inverse computation program of a given program. The second method is a transformational one whose output is compatible with the given specification and program. In these studies, we employ term rewriting systems as models to represent functions, programs and specifications.

**Inverse Compiler:** We are interested in a method for automatically generating an inverse program for a given program. Several inverse interpreters, such as unification algorithms, narrowing and "Inverse Algorithm" have been studied so far. For any program P and any value $v$ these inverse interpreters can search all solutions $(v_1, \ldots, v_n)$ such that $\mathtt{P}(v_1, \ldots, v_n) = v$. This is, however, difficult to combine with other programs: therefore, instead of inverse interpreters, we present an inverse compiler. Given a program P, our inverse compiler produces a term rewriting system that defines the inverse program $\mathtt{P}^{-1}$ which can be combined with other programs. Using this term rewriting system, we compute the inverse image of any data term $v$ with respect to P as the set of data terms of $\mathtt{P}^{-1}$. We also study a strategy to compute generated term rewriting systems efficiently.




**Program Generation from Quantified Equational Specifications:** In our transformational approach, we model specifications and programs as quantified equational formulas, and we realize the transformation based on quantified equational logic. Program generation succeeds if the output formulas represent a term rewriting system. This transformation is sound; that is, the resulted formulas logically imply the original formulas, which are representations of the given specification and program.



# Inverse Compiler

<u>What is the inverse computation of a program P?</u>

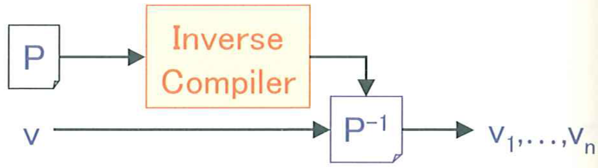
Find all tuples $(v_1,...,v_n)$ satisfying $P(v_1,...,v_n) = v$, from $v$.

Applications: solving equations, etc.

<u>Inverse Interpreter</u>


■ All solutions can be obtained.

<u>Inverse Compiler</u> (Our work)



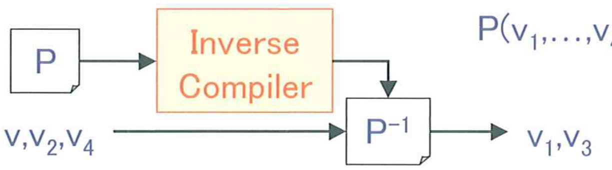

■ $P^{-1}$ is incorporable with other programs.

■ Easy to prove correctness and termination.

■ Possible to analyze properties of $P^{-1}$ from the properties of P.

# Extension of Inverse Compiler

<u>Partial Inverse Computation</u>




$P(v_1,...,v_4) = v$

<u>Ex.</u> addition and subtraction.

$$\begin{cases} 0 + y \to y \\ s(x) + y \to s(x+y) \end{cases} \Rightarrow \begin{cases} y - y \to 0 \\ s(z) - y \to s(x) \quad \text{if } z - y \to x \\ (x+y) - y \to x \end{cases}$$

given

<u>Application: program transformation</u>

$$\begin{cases} gcd(x+y, y) \to gcd(x,y) \\ gcd(x,y) \to gcd(y,x) \quad \text{if } x < y \\ gcd(x,0) \to x \end{cases} \Rightarrow \begin{cases} gcd(z, y) \to gcd(x,y) \quad \text{if } z - y \to x \\ gcd(x,y) \to gcd(y,x) \quad \text{if } x < y \\ gcd(x,0) \to x \end{cases}$$

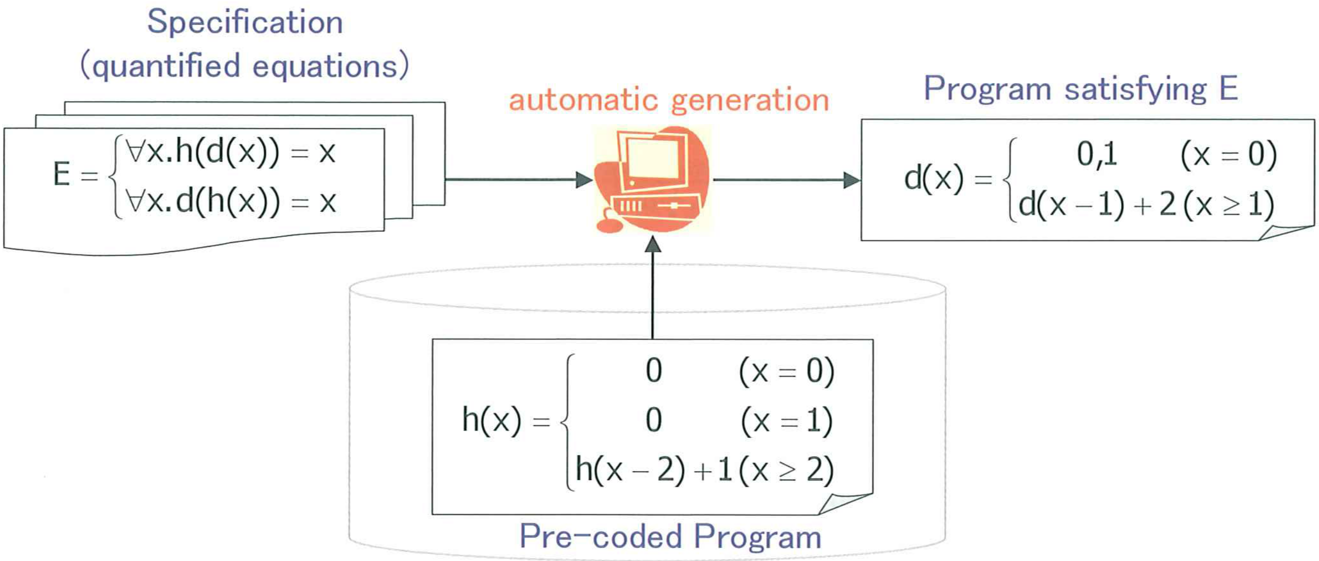
$gcd(s^4(0), s^2(0)) \not\xrightarrow{*} s^2(0)$ Not computable!

$gcd(s^4(0), s^2(0)) \xrightarrow{*} s^2(0)$ Computable!

is not an instance of x+y.

# Automatic Program Generation

## Our Purpose




Towards

- Reliable programs
- Efficient development

# Example of Transformation

## Transformation Rules

**Decomposition:**

$$\frac{E \cup \{\Gamma\langle C[t] \simeq s\rangle\}; R}{E \cup \{\Gamma\langle \exists x.(t \approx x \wedge C[x] \approx s)\rangle\}; R} \text{ if } \begin{array}{l} C[\,] \not\equiv \Box, \\ x \in \mathcal{X} - \mathcal{V}ar(\Gamma\langle C[t] \simeq s\rangle) \end{array}$$

**Composition:**

$$\frac{E \cup \{\Gamma\langle \exists x.(x \simeq t \wedge U)\rangle\}; R}{E \cup \{\Gamma\langle U\sigma\rangle\}; R} \text{ if } \sigma = \{x \mapsto t\}$$

**Expansion:**

$$\frac{E \cup \{\Gamma\langle \exists x.U\rangle\}; R}{E \cup \{\Gamma\langle \bigvee_i \exists\overrightarrow{y_i}.V_i\rangle\}; R} \text{ if } \begin{array}{l} \sigma_i = \{x \mapsto t_i\},\ \bigcup_i \{t_i\} : R\text{-covering set}, \\ \{\overrightarrow{y_i}\} = \mathcal{V}ar(t_i),\ U\sigma_i \to_{E \cup E_R} V_i \end{array}$$

**Deduction:**

$$\frac{E \cup \{\Gamma\langle \forall x.U\rangle\}; R}{E \cup \{\Gamma\langle \bigwedge_i \forall\overrightarrow{y_i}.V_i\rangle\}; R} \text{ if } \begin{array}{l} \sigma_i = \{x \mapsto t_i\},\ \bigcup_i \{t_i\} : R\text{-covering set}, \\ \{\overrightarrow{y_i}\} = \mathcal{V}ar(t_i),\ U\sigma_i \to_{E \cup E_R} V_i \end{array}$$

**Variable-Elimination:**

$$\frac{E \cup \{\Gamma\langle \forall x.(\bigvee_i \exists\overrightarrow{y_i}.(x \simeq t_i \wedge U_i))\rangle\}; R}{E \cup \{\Gamma\langle \bigwedge_i \forall\overrightarrow{y_i}.U_i\sigma_i\rangle\}; R} \text{ if } \begin{array}{l} \sigma_i = \{x \mapsto t_i\}, \\ \bigcup_i \{t_i\} : R\text{-strongly covering set}, \\ \{\overrightarrow{y_i}\} = \mathcal{V}ar(t_i) \end{array}$$

## Transformation Sequence

$$E_1; R_1 \equiv \left\{\begin{array}{c} \forall x.\,\langle d(h(x)) \approx x\rangle, \\ \forall x.h(d(x)) \approx x \end{array}\right\}; \left\{\begin{array}{c} h(0) \to 0, h(S(0)) \to 0, \\ h(S^2(x)) \to S(h(x)) \end{array}\right\} \Rightarrow_{\mathcal{G}}^{\mathbf{Dec}} \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \forall x.\,\langle h(d(x)) \approx x\rangle \end{array}\right\}; R_1 \Rightarrow_{\mathcal{G}}^{\mathbf{Dec}} \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \forall x.\,\langle \exists y.(d(x) \approx y \wedge h(y) \approx x)\rangle \end{array}\right\}; \underline{R_1}$$

$$\Rightarrow_{\mathcal{G}}^{\mathbf{Exp}} \left\{\begin{array}{c} \underline{\forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x)}, \\ \forall x. \left(\begin{array}{cc} & d(x) \approx 0 \wedge 0 \approx x \\ \vee & d(x) \approx S(0) \wedge 0 \approx x \\ \vee & \langle \exists z.(d(x) \approx S^2(z) \wedge S(h(z)) \approx x)\rangle \end{array}\right) \end{array}\right\}; R_1 \Rightarrow_{\mathcal{G}}^{\mathbf{Exp}} \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \forall x. \left(\begin{array}{cc} & d(x) \approx 0 \wedge 0 \approx x \\ \vee & d(x) \approx S(0) \wedge 0 \approx x \\ \vee & \exists z.(d(x) \approx S^2(z) \wedge \exists u.(S(u) \approx x \wedge d(u) \approx z)) \end{array}\right) \end{array}\right\}; R_1$$

$$\cong \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \left\langle \forall x. \left(\begin{array}{cc} & (d(x) \approx 0 \vee d(x) \approx S(0)) \wedge x \approx 0 \\ \vee & \exists u.\exists z.(d(x) \approx S^2(z) \wedge x \approx S(u) \wedge d(u) \approx z) \end{array}\right)\right\rangle \end{array}\right\}; R_1 \Rightarrow_{\mathcal{G}}^{\mathbf{V\text{-}Eli}} \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \left(\begin{array}{cc} & (d(0) \approx 0 \vee d(0) \approx S(0)) \\ \wedge & \forall u.\,\langle \exists z.(d(S(u)) \approx S^2(z) \wedge d(u) \approx z)\rangle \end{array}\right) \end{array}\right\}; R_1$$

$$\Rightarrow_{\mathcal{G}}^{\mathbf{Com}} \left\{\begin{array}{c} \forall x.\exists y.(h(x) \approx y \wedge d(y) \approx x), \\ \left(\begin{array}{cc} & (d(0) \approx 0 \vee d(0) \approx S(0)) \\ \wedge & \forall u.d(S(u)) \approx S^2(d(u)) \end{array}\right) \end{array}\right\}; R_1 \equiv E_2; R_1$$

$$R_2 \equiv \left\{\begin{array}{c} d(0) \to 0, \\ d(0) \to 1, \\ d(S(x)) \to S^2(d(x)) \end{array}\right\}$$