Proving Equivalence of Imperative Programs via Constrained Rewriting Induction

Cynthia Kop (University of Copenhagen)

joint work with
Carsten Fuhs (Birkbeck, U of London) and Naoki Nishida (U Nagoya)

Danish Static Analysis Symposium, Odense
19 August 2016

Overview

Overview

C Programming Course in Nagoya

C Programming Course in Nagoya

- $\pm$70 students every year (of whom 60 active)
- 3 programming exercises every week
- $\Longrightarrow$ 180^{+} exercises to grade every week for a full semester
- student programs can be horrible

Exercise: write a function that calculates $\Sigma_{k=1}^{n} k$.

```
int sum(int x) {
  int i = 0, z = 0;
  for (i = 0; i <= x; i++)
    z += i;
  return z;
}
```

```
int sum( int n ){
  int cnt;
  int data;
  if(n < 0){
    return 0;
  }
  for(cnt = 1;cnt <= n;cnt++){
    data = data + cnt;
  }
  return data;
}
```

```
int sum(int n)
{
  if ( n<=0 ) {
    return 0;
  } else {
    return (n*(n+1)/2);
  }
}
```

```
int sum(int x) {
  int i, j, z;
  z = 0;
  for (i = 0; i <= x; i++)
    for (j = 0; j < i; j++)
      z++;
    return z;
}
```

Solutions

Solutions

- hire some teaching assistants!

Solutions

- hire some teaching assistants!
- automate the marking

Solutions

- hire some teaching assistants!
- automate the marking

Automated Program Testing

Automated Program Testing

- run automatic tests

Automated Program Testing

- run automatic tests
- prove that programs are correct!

Automated Program Testing

- run automatic tests
- prove that programs are correct!

Automated Program Testing

- run automatic tests
- prove that programs are correct!
  - we are experts in term rewriting

Automated Program Testing

- run automatic tests
- prove that programs are correct!
  - we are experts in term rewriting
  - $\Rightarrow$ convert C programs to term rewriting systems!

Automated Program Testing

- run automatic tests
- prove that programs are correct!
  - we are experts in term rewriting
  - $\Rightarrow$ convert C programs to term rewriting systems!
  - $\Rightarrow$ reason about those TRSs instead!

Overview

What's Term Rewriting?

What's Term Rewriting?

Syntactic approach for reasoning in equational first-order logic

What's Term Rewriting?

Syntactic approach for reasoning in equational first-order logic

Core functional programming language without many restrictions (and features) of "real" FP:

What's Term Rewriting?

Syntactic approach for reasoning in equational first-order logic

Core functional programming language without many restrictions (and features) of "real" FP:

- first-order (usually)
- no fixed evaluation strategy
- no fixed order of rules to apply (Haskell: top to bottom)
- untyped
- no pre-defined data structures (integers, arrays, ...)

Summing up Natural Numbers

Numbers: $0,\ \mathsf{s}(0),\ \mathsf{s}(\mathsf{s}(0)),\ \ldots$

Summing up Natural Numbers

Numbers: $0,\ \mathsf{s}(0),\ \mathsf{s}(\mathsf{s}(0)),\ \dots$

Rules:

$$\begin{array}{rcl} \mathsf{sum}(0) & \to & 0 \\ \mathsf{sum}(\mathsf{s}(x)) & \to & \mathsf{plus}(\mathsf{s}(x), \mathsf{sum}(x)) \\ \mathsf{plus}(0, y) & \to & y \\ \mathsf{plus}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

Summing up Natural Numbers

Numbers: 0, s(0), s(s(0)), ...

Rules:

$$\begin{array}{rcl} \mathsf{sum}(0) & \to & 0 \\ \mathsf{sum}(\mathsf{s}(x)) & \to & \mathsf{plus}(\mathsf{s}(x), \mathsf{sum}(x)) \\ \mathsf{plus}(0, y) & \to & y \\ \mathsf{plus}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

Then e.g. we can compute $1 + 1 = 2$ as

$$\mathsf{plus}(\mathsf{s}(0), \mathsf{s}(0)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{plus}(0, \mathsf{s}(0))) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0))$$

Summing up Natural Numbers

Numbers: 0, s(0), s(s(0)), ...

Rules:

$$\begin{array}{rcl} \mathsf{sum}(0) & \to & 0 \\ \mathsf{sum}(\mathsf{s}(x)) & \to & \mathsf{plus}(\mathsf{s}(x), \mathsf{sum}(x)) \\ \mathsf{plus}(0, y) & \to & y \\ \mathsf{plus}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

Then e.g. we can compute $1 + 1 = 2$ as

$$\mathsf{plus}(\mathsf{s}(0), \mathsf{s}(0)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{plus}(0, \mathsf{s}(0))) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0))$$

Integer arithmetic possible with more complex recursive rules.

Summing up Natural Numbers

Numbers: $0,\ \mathsf{s}(0),\ \mathsf{s}(\mathsf{s}(0)),\ \ldots$

Rules:

$$\begin{array}{rcl} \mathsf{sum}(0) & \to & 0 \\ \mathsf{sum}(\mathsf{s}(x)) & \to & \mathsf{plus}(\mathsf{s}(x), \mathsf{sum}(x)) \\ \mathsf{plus}(0, y) & \to & y \\ \mathsf{plus}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{plus}(x, y)) \end{array}$$

Then e.g. we can compute $1 + 1 = 2$ as

$$\mathsf{plus}(\mathsf{s}(0), \mathsf{s}(0)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{plus}(0, \mathsf{s}(0))) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0))$$

Integer arithmetic possible with more complex recursive rules.

But: We want to do **program analysis**. Should we really throw away domain knowledge about built-in data structures?!

What's Constrained Term Rewriting?

Term rewriting "with batteries included"

- first-order
- no fixed evaluation strategy
- no fixed order of rules to apply

What's Constrained Term Rewriting?

Term rewriting "with batteries included"

- first-order
- no fixed evaluation strategy
- no fixed order of rules to apply
- typed

What's Constrained Term Rewriting?

Term rewriting "with batteries included"

- first-order
- no fixed evaluation strategy
- no fixed order of rules to apply
- typed
- with pre-defined data structures (integers, arrays, bitvectors, ...), usually from SMT-LIB theories (SMT: SAT Modulo Theories)

What's Constrained Term Rewriting?

Term rewriting "with batteries included"

- first-order
- no fixed evaluation strategy
- no fixed order of rules to apply
- typed
- with pre-defined data structures (integers, arrays, bitvectors, ...), usually from SMT-LIB theories (SMT: SAT Modulo Theories)
- rewrite rules with SMT constraints

What's Constrained Term Rewriting?

Term rewriting "with batteries included"

- first-order
- no fixed evaluation strategy
- no fixed order of rules to apply
- typed
- with pre-defined data structures (integers, arrays, bitvectors, ...), usually from SMT-LIB theories (SMT: SAT Modulo Theories)
- rewrite rules with SMT constraints

$\Rightarrow$ Term rewriting + SMT solving for automated reasoning

Integer Summation

$$\begin{array}{lcll} \mathsf{sum}(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0] \end{array}$$

Integer Summation

$$\begin{array}{lcll} \mathsf{sum}(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0] \end{array}$$

$\mathsf{sum}(2)$

Integer Summation

$$\begin{array}{lcll} \mathsf{sum}(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0] \end{array}$$

$$\begin{array}{ll} & \mathsf{sum}(2) \\ \to & 2 + \mathsf{sum}(2-1) \end{array}$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x - 1) & [x > 0]
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{sum}(2) \\
\to & 2 + \mathsf{sum}(2 - 1) \\
\to & 2 + \mathsf{sum}(1)
\end{array}
$$

Integer Summation

$$\begin{array}{lcll} \mathsf{sum}(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0] \end{array}$$

$$\begin{array}{ll} & \mathsf{sum}(2) \\ \to & 2 + \mathsf{sum}(2-1) \\ \to & 2 + \mathsf{sum}(1) \\ \to & 2 + (1 + \mathsf{sum}(1-1)) \end{array}$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0]
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{sum}(2) \\
\to & 2 + \mathsf{sum}(2-1) \\
\to & 2 + \mathsf{sum}(1) \\
\to & 2 + (1 + \mathsf{sum}(1-1)) \\
\to & 2 + (1 + \mathsf{sum}(0))
\end{array}
$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0]
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{sum}(2) \\
\to & 2 + \mathsf{sum}(2-1) \\
\to & 2 + \mathsf{sum}(1) \\
\to & 2 + (1 + \mathsf{sum}(1-1)) \\
\to & 2 + (1 + \mathsf{sum}(0)) \\
\to & 2 + (1 + 0)
\end{array}
$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0]
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{sum}(2) \\
\to & 2 + \mathsf{sum}(2-1) \\
\to & 2 + \mathsf{sum}(1) \\
\to & 2 + (1 + \mathsf{sum}(1-1)) \\
\to & 2 + (1 + \mathsf{sum}(0)) \\
\to & 2 + (1 + 0) \\
\to & 2 + 1
\end{array}
$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x - 1) & [x > 0]
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{sum}(2) \\
\to & 2 + \mathsf{sum}(2 - 1) \\
\to & 2 + \mathsf{sum}(1) \\
\to & 2 + (1 + \mathsf{sum}(1 - 1)) \\
\to & 2 + (1 + \mathsf{sum}(0)) \\
\to & 2 + (1 + 0) \\
\to & 2 + 1 \\
\to & 3
\end{array}
$$

Integer Summation

$$
\begin{array}{lcll}
\mathsf{sum}(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}(x) & \to & x + \mathsf{sum}(x-1) & [x > 0]
\end{array}
$$

- $\mathcal{F}_{terms} = \{\mathsf{sum}\} \cup \{\mathsf{n} \mid n \in \mathbb{Z}\}$
- $\mathcal{F}_{theory} =$
  $\{+, -, \geq, >, \wedge, \mathsf{true}, \mathsf{false}\} \cup \{\mathsf{n} \mid n \in \mathbb{Z}\}$
- Values: $\mathsf{true}, \mathsf{false}, 0, 1, 2, 3, \ldots, -1, -2, \ldots$
- Interpretation: addition, minus, etc.

Bitvector Summation

$$\begin{array}{lcll} \mathsf{sum}(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}(x) & \to & x + \mathsf{sum}(x - 1) & [x > 0] \end{array}$$

- $\mathcal{F}_{terms} = \{\mathsf{sum}\} \cup \{\mathsf{n} \mid n \in \mathbb{Z} \land 0 \leq n < 256\}$
- $\mathcal{F}_{theory} =$
  $\{+, -, \geq, >, \land, \mathsf{true}, \mathsf{false}\} \cup \{\mathsf{n} \mid n \in \mathbb{Z} \land 0 \leq n < 256\}$
- Values: $\mathsf{true}, \mathsf{false}, 0, 1, 2, 3, \ldots, 255$
- Interpretation: addition, minus, etc. modulo 256

Array Summation

$$\begin{array}{lcll} \mathsf{sum}(a,x) & \to & 0 & [x < 0] \\ \mathsf{sum}(a,x) & \to & \mathsf{select}(a,x) + \mathsf{sum}(a, x-1) & [x \geq 0] \end{array}$$

- $\mathcal{F}_{terms} = \{\mathsf{sum}\} \cup \{\mathsf{n} : \mathsf{int} \mid n \in \mathbb{Z}\} \cup \{\mathsf{a} : \mathsf{iarr} \mid n \in \mathbb{Z}^*\}$
- $\mathcal{F}_{theory} =$
  $\{+, -, \geq, >, \wedge, \mathsf{select}, \mathsf{true}, \mathsf{false}\} \cup \{\mathsf{n} \mid n \in \mathbb{Z}\} \cup$
  $\{\mathsf{a} : \mathsf{iarr} \mid a \in \mathbb{Z}^*\}$
- Values:
  $\mathsf{true}, \mathsf{false}, 0, 1, -1, 2, -2, \ldots, (), (0), (1), \ldots, (0,0), \ldots$

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

- work much like normal term rewrite systems:

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

- work much like normal term rewrite systems:
  - terms are expressions built from function symbols
  - rules are used to rewrite terms

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

- work much like normal term rewrite systems:
  - terms are expressions built from function symbols
  - rules are used to rewrite terms
- can handle integers, arrays, bitvectors, ...

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

- work much like normal term rewrite systems:
  - terms are expressions built from function symbols
  - rules are used to rewrite terms
- can handle integers, arrays, bitvectors, ...
- no predefined behaviour!

Logically Constrained Term Rewriting Systems [Kop and Nishida, 2013]

- work much like normal term rewrite systems:
  - terms are expressions built from function symbols
  - rules are used to rewrite terms
- can handle integers, arrays, bitvectors, ...
- no predefined behaviour!
- are flexible enough to faithfully model (many) real-world programs

Overview

Factorial

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

Factorial

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{u}_1(x) & \\
\mathsf{u}_1(x) & \to & \mathsf{u}_2(x, 1, 1) & \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{u}_3(x, z, i) & [i \leq x] \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{u}_4(x, z, i) & [\neg(i \leq x)] \\
\mathsf{u}_3(x, z, i) & \to & \mathsf{u}_2(x, z * i, i + 1) & \\
\mathsf{u}_4(x, z, i) & \to & z &
\end{array}
$$

Factorial

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{u}_2(x, 1, 1) & \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{u}_2(x, z * i, i + 1) & [i \leq x] \\
\mathsf{u}_2(x, z, i) & \to & z & [\neg(i \leq x)]
\end{array}
$$

Factorial

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{u}_2(x, 1, 1) & \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{u}_2(x, z * i, i + 1) & [i \leq x] \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{return}(z) & [\neg(i \leq x)]
\end{array}
$$

Division by Zero

```
boolean divides(int x, int y) {
  return x % y == 0;
}
```

Division by Zero

```
boolean divides(int x, int y) {
  return x % y == 0;
}
```

$$\mathsf{divides}(x, y) \quad \to \quad \mathsf{return}(x \;\mathsf{mod}\; y = 0)$$

Division by Zero

```
boolean divides(int x, int y) {
  return x % y == 0;
}
```

$$\begin{array}{lcll} \mathsf{divides}(x, y) & \to & \mathsf{return}(x \ \mathsf{mod}\ y = 0) & [y \neq 0] \\ \mathsf{divides}(x, y) & \to & \mathsf{error} & [y = 0] \end{array}$$

Division by Zero

```
boolean divides(int x, int y) {
  return x % y == 0;
}
```

$$\begin{array}{lcll} \mathsf{divides}(x,y) & \to & \mathsf{return}(x \bmod y = 0) & [y \neq 0] \\ \mathsf{divides}(x,y) & \to & \mathsf{error} & [y = 0] \end{array}$$

(defining $x \bmod 0 = 0$)

Integer Overflow

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

$$\begin{array}{rcll} \mathsf{fact}(x) & \to & \mathsf{u}_2(x, 1, 1) & \\ \mathsf{u}_2(x, z, i) & \to & \mathsf{u}_2(x, z * i, i + 1) & [i \leq x] \\ & & & \\ \mathsf{u}_2(x, z, i) & \to & \mathsf{return}(z) & [\neg(i \leq x)] \end{array}$$

Integer Overflow

```
int fact(int x) {
  int z = 1;
  for (int i = 1; i <= x; i++)
    z *= i;
  return z;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{u}_2(x, 1, 1) & \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{u}_2(x, z * i, i + 1) & [i \leq x \wedge z * i < 256 \wedge i + 1 < 256] \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{error} & [i \leq x \wedge (z * i \geq 256 \vee i + 1 \geq 256)] \\
\mathsf{u}_2(x, z, i) & \to & \mathsf{return}(z) & [\neg(i \leq x)]
\end{array}
$$

Recursion

```
int fact(int x) {
  if (x > 0) return x * fact(x-1);
  else return 1;
}
```

Recursion

```
int fact(int x) {
  if (x > 0) return x * fact(x-1);
  else return 1;
}
```

$$\begin{array}{lcll} \mathsf{fact}(x) & \to & x * \mathsf{fact}(x-1) & [x > 0] \\ \mathsf{fact}(x) & \to & 1 & [\neg(x > 0)] \end{array}$$

Recursion

```
int fact(int x) {
  if (x > 0) return x * fact(x-1);
  else return 1;
}
```

$$\begin{array}{lcll} \mathsf{fact}(x) & \to & x * \mathsf{fact}(x-1) & [x > 0] \\ \mathsf{fact}(x) & \to & \mathsf{return}(1) & [\neg(x > 0)] \end{array}$$

Recursion

```
int fact(int x) {
  if (x > 0) return x * fact(x-1);
  else return 1;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{helper}(x, \mathsf{fact}(x-1)) & [x > 0] \\
\mathsf{fact}(x) & \to & \mathsf{return}(1) & [\neg(x > 0)] \\
\mathsf{helper}(x, \mathsf{return}(y)) & \to & \mathsf{return}(x * y) &
\end{array}
$$

Recursion with Errors

```
int fact(int x) {
  if (x > 0) return x * fact(x-1);
  else return 1;
}
```

$$
\begin{array}{rcll}
\mathsf{fact}(x) & \to & \mathsf{helper}(x, \mathsf{fact}(x-1)) & [x > 0] \\
\mathsf{fact}(x) & \to & \mathsf{return}(1) & [\neg(x > 0)] \\
\mathsf{helper}(x, \mathsf{return}(y)) & \to & x * y & [x * y < 256] \\
\mathsf{helper}(x, \mathsf{return}(y)) & \to & \mathsf{error} & [x * y \geq 256] \\
\mathsf{helper}(x, \mathsf{error}) & \to & \mathsf{error} &
\end{array}
$$

Global Variables

```
int best;
int up(int x) {
  if (x > best) {
    best = x;
    return 1;
  }
  return 0;
}
```

```
void main() {
  while(1) {
    int k = input();
    up(k);
    if (!k) break;
  }
}
```

$$
\begin{array}{rcll}
\mathsf{up}(b,x) & \to & \mathsf{return_{up}}(x,1) & [x > b] \\
\mathsf{up}(b,x) & \to & \mathsf{return_{up}}(b,0) & [\neg(x > b)] \\
\mathsf{main}(b) & \to & \mathsf{u}_1(b,inp) & \\
\mathsf{u}_1(b,k) & \to & \mathsf{u}_2(k,\mathsf{up}(b,k)) & \\
\mathsf{u}_2(k,\mathsf{return_{up}}(b',i)) & \to & \mathsf{return_{main}}(b') & [\neg(k \neq 0)] \\
\mathsf{u}_2(k,\mathsf{return_{up}}(b',i)) & \to & \mathsf{u}_1(b',inp) & [k \neq 0]
\end{array}
$$

Statically Allocated Arrays

```
void strcpy(char goal[], char original[]) {
  int i = 0;
  for (; original[i]; i++) goal[i] = original[i];
  goal[i] = 0;
}
```

Statically Allocated Arrays

```
void strcpy(char goal[], char original[]) {
  int i = 0;
  for (; original[i]; i++) goal[i] = original[i];
  goal[i] = 0;
}
```

$$
\begin{array}{rcl}
\mathsf{strcpy}(x,y) & \to & \mathsf{v}(x,y,\mathsf{0}) \\
\mathsf{v}(x,y,i) & \to & \mathsf{w}(x,y,i) \quad [\mathsf{select}(y,i) = 0] \\
\mathsf{v}(x,y,i) & \to & \mathsf{v}(\mathsf{store}(x,i,\mathsf{select}(y,i)),y,i+\mathsf{1}) \\
 & & \qquad\qquad\qquad\qquad [\mathsf{select}(y,i) \neq 0] \\
\mathsf{w}(x,y,i) & \to & \mathsf{return}(\mathsf{store}(x,i,\mathsf{0}))
\end{array}
$$

Statically Allocated Arrays

```
void strcpy(char goal[], char original[]) {
  int i = 0;
  for (; original[i]; i++) goal[i] = original[i];
  goal[i] = 0;
}
```

$$
\begin{array}{rcl}
\mathsf{strcpy}(x, y) & \to & \mathsf{v}(x, y, 0) \\
\mathsf{v}(x, y, i) & \to & \mathsf{w}(x, y, i) \quad [0 \leq i < \mathsf{size}(y) \wedge \mathsf{select}(y, i) = 0] \\
\mathsf{v}(x, y, i) & \to & \mathsf{v}(\mathsf{store}(x, i, \mathsf{select}(y, i)), y, i + 1) \quad [0 \leq i < \\
 & & \qquad \mathsf{size}(x) \wedge i < \mathsf{size}(y) \wedge \mathsf{select}(y, i) \neq 0] \\
\mathsf{w}(x, y, i) & \to & \mathsf{return}(\mathsf{store}(x, i, 0)) \quad [0 \leq i < \mathsf{size}(x)]
\end{array}
$$

Statically Allocated Arrays

```
void strcpy(char goal[], char original[]) {
  int i = 0;
  for (; original[i]; i++) goal[i] = original[i];
  goal[i] = 0;
}
```

$$
\begin{array}{rcl}
\mathsf{strcpy}(x, y) & \to & \mathsf{v}(x, y, 0) \\
\mathsf{v}(x, y, i) & \to & \mathsf{w}(x, y, i) \quad [0 \le i < \mathsf{size}(y) \wedge \mathsf{select}(y, i) = 0] \\
\mathsf{v}(x, y, i) & \to & \mathsf{v}(\mathsf{store}(x, i, \mathsf{select}(y, i)), y, i + 1) \quad [0 \le i < \\
 & & \qquad \mathsf{size}(x) \wedge i < \mathsf{size}(y) \wedge \mathsf{select}(y, i) \neq 0] \\
\mathsf{w}(x, y, i) & \to & \mathsf{return}(\mathsf{store}(x, i, 0)) \quad [0 \le i < \mathsf{size}(x)] \\
\mathsf{v}(x, y, i) & \to & \mathsf{error} \quad [i < 0 \vee i \ge \mathsf{size}(y) \vee \\
 & & \qquad (\mathsf{select}(y, i) \neq 0 \wedge i \ge \mathsf{size}(x))] \\
\mathsf{w}(x, y, i) & \to & \mathsf{error} \quad [i < 0 \vee i \ge \mathsf{size}(x)]
\end{array}
$$

Dynamically Allocated Arrays

- model memory as a sequence of integer sequences
- a dynamic array is a pair (index, offset)
- `int *a = new int[10];`
  $\Rightarrow$ $\mathsf{u}_1(mem) \to$
  $\mathsf{u}_2(\mathsf{add}(mem, x), \mathsf{pair}(\mathsf{size}(mem), 0))\ [\mathsf{size}(x) = 10]$
- `int k = a[3];`
  $\Rightarrow$ $\mathsf{u}_2(mem, \mathsf{pair}(x, y)) \to$
  $\mathsf{u}_3(mem, \mathsf{pair}(x, y), \mathsf{select}(\mathsf{select}(mem, x), y + 3))$
  $[0 \leq y + 3 < \mathsf{size}(\mathsf{select}(mem, x))]$
  (Note: $\mathsf{select}(mem, a)$ returns () if a is out of bound.)
- `int *b = a + 1;`
  $\Rightarrow$ $\mathsf{u}_3(mem, \mathsf{pair}(x, y), k) \to$
  $\mathsf{u}_4(mem, \mathsf{pair}(x, y), k, \mathsf{pair}(x, y + 1))$

Overview

What is Equivalence for LCTRSs?

Teacher's code:

$$
\begin{array}{rcll}
\mathsf{sum}_1(x) & \to & \mathsf{return}(0) & [x \leq 0] \\
\mathsf{sum}_1(x) & \to & \mathsf{helper}(x, \mathsf{sum}_1(x-1)) & [x > 0] \\
\mathsf{helper}(x, \mathsf{return}(y)) & \to & \mathsf{helper}(x+y) &
\end{array}
$$

What is Equivalence for LCTRSs?

Teacher's code:

$$\begin{array}{rcll} \mathsf{sum}_1(x) & \to & \mathsf{return}(0) & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & \mathsf{helper}(x, \mathsf{sum}_1(x-1)) & [x > 0] \\ \mathsf{helper}(x, \mathsf{return}(y)) & \to & \mathsf{helper}(x+y) & \end{array}$$

Student's code:

$$\begin{array}{rcll} \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & \mathsf{return}(z) & [\neg(i \leq x)] \end{array}$$

What is Equivalence for LCTRSs?

Teacher's code:

$$\begin{array}{rcll} \mathsf{sum}_1(x) & \to & \mathsf{return}(0) & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & \mathsf{helper}(x, \mathsf{sum}_1(x-1)) & [x > 0] \\ \mathsf{helper}(x, \mathsf{return}(y)) & \to & \mathsf{helper}(x + y) & \end{array}$$

Student's code:

$$\begin{array}{rcll} \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & \mathsf{return}(z) & [\neg(i \leq x)] \end{array}$$

Query: $\mathsf{sum}_1(x) \leftrightarrow^* \mathsf{sum}_2(x)$ for all x?

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1 \; [\varphi_1], \; \ldots, \; s_n \approx t_n \; [\varphi_n]$

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1\ [\varphi_1],\ \ldots,\ s_n \approx t_n\ [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x)\ [\mathsf{true}]\}$

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1 \ [\varphi_1], \ \dots, \ s_n \approx t_n \ [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x) \ [\mathsf{true}]\}$
- set of rewrite rules $\mathcal{R}$

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1\ [\varphi_1],\ \ldots,\ s_n \approx t_n\ [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x)\ [\mathsf{true}]\}$
- set of rewrite rules $\mathcal{R}$

Want to prove:

for all constructor ground substitutions $\gamma_1, \ldots, \gamma_n$ compatible with $\varphi_1, \ldots, \varphi_n$: each $s_i\gamma_i \leftrightarrow^*_{\mathcal{R}} t_i\gamma_i$.

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1\ [\varphi_1],\ \ldots,\ s_n \approx t_n\ [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x)\ [\mathsf{true}]\}$
- set of rewrite rules $\mathcal{R}$

Want to prove:
for all constructor ground substitutions $\gamma_1, \ldots, \gamma_n$ compatible with $\varphi_1, \ldots, \varphi_n$: each $s_i\gamma_i \leftrightarrow^*_{\mathcal{R}} t_i\gamma_i$.

Requirements:

- termination of $\to_{\mathcal{R}}$ (to perform induction)

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1\ [\varphi_1],\ \ldots,\ s_n \approx t_n\ [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x)\ [\mathsf{true}]\}$
- set of rewrite rules $\mathcal{R}$

Want to prove:
for all constructor ground substitutions $\gamma_1, \ldots, \gamma_n$ compatible with $\varphi_1, \ldots, \varphi_n$: each $s_i\gamma_i \leftrightarrow^*_{\mathcal{R}} t_i\gamma_i$.

Requirements:

- termination of $\rightarrow_{\mathcal{R}}$ (to perform induction)
- sufficient completeness of $\rightarrow_{\mathcal{R}}$: evaluation "cannot get stuck" (for case analysis over variables by constructor terms)

Rewriting Induction

Given:

- set $\mathcal{E}$ of equations $s_1 \approx t_1 \; [\varphi_1], \; \ldots, \; s_n \approx t_n \; [\varphi_n]$
  $\Longrightarrow$ for example: $\mathcal{E} = \{\mathsf{sum1}(x) \approx \mathsf{sum2}(x) \; [\mathsf{true}]\}$
- set of rewrite rules $\mathcal{R}$

Want to prove:
for all constructor ground substitutions $\gamma_1, \ldots, \gamma_n$ compatible with $\varphi_1, \ldots, \varphi_n$: each $s_i\gamma_i \leftrightarrow^*_{\mathcal{R}} t_i\gamma_i$.

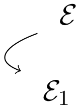
Requirements:

- termination of $\to_{\mathcal{R}}$ (to perform induction)
- sufficient completeness of $\to_{\mathcal{R}}$: evaluation "cannot get stuck" (for case analysis over variables by constructor terms)
- if we want $s_i\gamma_i \to * \leftarrow t_i\gamma_i$: confluence of $\to_{\mathcal{R}}$

Rewriting Induction

$$\mathcal{E}$$

Rewriting Induction



Rewriting Induction

$\mathcal{E}$

$\mathcal{E}_1$

$\mathcal{E}_2$

Rewriting Induction

$\mathcal{E}$

$\mathcal{E}_1$

$\mathcal{E}_2$

$\mathcal{E}_3$

Rewriting Induction

$$\mathcal{E}$$

$$\mathcal{E}_1$$

$$\mathcal{E}_2$$

$$\mathcal{E}_3$$

$$\mathcal{E}_4$$

Rewriting Induction

$$(\mathcal{E}, \emptyset)$$

$$\downarrow$$

$$\mathcal{E}_1$$

$$\downarrow$$

$$\mathcal{E}_2$$

$$\downarrow$$

$$\mathcal{E}_3$$

$$\downarrow$$

$$\mathcal{E}_4$$

Rewriting Induction

$$(\mathcal{E}, \emptyset)$$

$$\rightsquigarrow (\mathcal{E}_1, \mathcal{H}_1)$$

$$\rightsquigarrow \mathcal{E}_2$$

$$\rightsquigarrow \mathcal{E}_3$$

$$\rightsquigarrow \mathcal{E}_4$$

Rewriting Induction

$$(\mathcal{E}, \emptyset)$$
$$\curvearrowleft$$
$$(\mathcal{E}_1, \mathcal{H}_1)$$
$$\curvearrowleft$$
$$(\mathcal{E}_2, \mathcal{H}_2)$$
$$\curvearrowleft$$
$$\mathcal{E}_3$$
$$\curvearrowleft$$
$$\mathcal{E}_4$$

Rewriting Induction

$$(\mathcal{E}, \emptyset)$$

$$\curvearrowright$$

$$(\mathcal{E}_1, \mathcal{H}_1)$$

$$\curvearrowright$$

$$(\mathcal{E}_2, \mathcal{H}_2)$$

$$\curvearrowright$$

$$(\mathcal{E}_3, \mathcal{H}_3)$$

$$\curvearrowright$$

$$\mathcal{E}_4$$

Rewriting Induction

$$(\mathcal{E}, \emptyset)$$

$$(\mathcal{E}_1, \mathcal{H}_1)$$

$$(\mathcal{E}_2, \mathcal{H}_2)$$

$$(\mathcal{E}_3, \mathcal{H}_3)$$

$$(\mathcal{E}_4, \mathcal{H}_4)$$

Rewriting Induction

$$
\begin{array}{c}
(\mathcal{E}, \emptyset) \\
\curvearrowright \\
(\mathcal{E}_1, \mathcal{H}_1) \\
\curvearrowright \\
(\mathcal{E}_2, \mathcal{H}_2) \\
\curvearrowright \\
(\mathcal{E}_3, \mathcal{H}_3) \\
\curvearrowright \\
(\mathcal{E}_4, \mathcal{H}_4) \\
\curvearrowright \\
(\emptyset, \mathcal{H}_4)
\end{array}
$$

Rewriting Induction

Three sets:

- $\mathcal{E}$ (equations, “the queries”)
- $\mathcal{R}$ (rules, “the program”)
- $\mathcal{H}$ (rules, “induction hypotheses”)

Rewriting Induction

Three sets:

- $\mathcal{E}$ (equations, "the queries")
- $\mathcal{R}$ (rules, "the program")
- $\mathcal{H}$ (rules, "induction hypotheses")

Initially: $\mathcal{E}$ given, $\mathcal{R}$ given, $\mathcal{H}$ empty

Rewriting Induction

Three sets:

- $\mathcal{E}$ (equations, "the queries")
- $\mathcal{R}$ (rules, "the program")
- $\mathcal{H}$ (rules, "induction hypotheses")

Initially: $\mathcal{E}$ given, $\mathcal{R}$ given, $\mathcal{H}$ empty

Proof steps: pairs $(\mathcal{E}, \mathcal{H}) \vdash (\mathcal{E}', \mathcal{H}')$ by several inference rules for $\vdash$

Rewriting Induction

Three sets:

- $\mathcal{E}$ (equations, "the queries")
- $\mathcal{R}$ (rules, "the program")
- $\mathcal{H}$ (rules, "induction hypotheses")

Initially: $\mathcal{E}$ given, $\mathcal{R}$ given, $\mathcal{H}$ empty

Proof steps: pairs $(\mathcal{E}, \mathcal{H}) \vdash (\mathcal{E}', \mathcal{H}')$ by several inference rules for $\vdash$

Invariant: $\to_{\mathcal{R} \cup \mathcal{H}}$ terminating

Rewriting Induction

Three sets:

- $\mathcal{E}$ (equations, "the queries")
- $\mathcal{R}$ (rules, "the program")
- $\mathcal{H}$ (rules, "induction hypotheses")

Initially: $\mathcal{E}$ given, $\mathcal{R}$ given, $\mathcal{H}$ empty

Proof steps: pairs $(\mathcal{E}, \mathcal{H}) \vdash (\mathcal{E}', \mathcal{H}')$ by several inference rules for $\vdash$

Invariant: $\to_{\mathcal{R} \cup \mathcal{H}}$ terminating

Goal: find derivation $(\mathcal{E}, \emptyset) \vdash^* (\emptyset, \mathcal{H})$

Then also $\leftrightarrow^*_{\mathcal{E}} \subseteq \leftrightarrow^*_{\mathcal{R} \cup \mathcal{H}} \subseteq \leftrightarrow^*_{\mathcal{R}}$ on ground terms:
Equations $\mathcal{E}$ are **inductive theorems** for $\mathcal{R}$

Simplification: definition

$$\frac{(\mathcal{E} \uplus \{s \simeq t\ [\varphi]\}, \mathcal{H})}{(\mathcal{E} \cup \{s' \approx t\ [\psi]\}, \mathcal{H})}$$

if $\quad s \approx t\ [\varphi] \quad \rightarrow_{\mathcal{R} \cup \mathcal{H}} \quad s' \approx t\ [\psi]$

Idea: Use the program or an induction hypothesis to simplify the query.

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x,y,z) \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1] \\ \}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x,y,z) \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1] \\ \}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x, y+1, z+y) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1] \}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x, y+1, z+y) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1] \}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x, y', z+y) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1]\}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x, y', z + y) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \wedge\, y' = y + 1]\}, \mathcal{H})$$

Simplification: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x,y',z') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \\ \wedge\, y' = y + 1 \wedge z' = z + y]\}, \mathcal{H})$$

Expansion: definition

$$\frac{(\mathcal{E} \uplus \{s \simeq t\ [\varphi]\}, \mathcal{H})}{(\mathcal{E} \cup \mathit{Expd}(s, t, \varphi, p), \mathcal{H} \cup \{s \to t\ [\varphi]\})}$$

if for every γ compatible with φ, $s_{|p}$ reduces and $\mathcal{R} \cup \mathcal{H} \cup \{s \to t\ [\varphi]\}$ is terminating

Expansion: definition

$$\frac{(\mathcal{E} \uplus \{s \simeq t\ [\varphi]\}, \mathcal{H})}{(\mathcal{E} \cup \mathit{Expd}(s, t, \varphi, p), \mathcal{H} \cup \{s \to t\ [\varphi]\})}$$

if for every γ compatible with φ, $s_{|p}$ reduces and $\mathcal{R} \cup \mathcal{H} \cup \{s \to t\ [\varphi]\}$ is terminating

$\mathit{Expd}(C[l']_p, t, \varphi, p)$ contains equations $C[r\gamma]_p \approx t\gamma\ [\varphi\gamma \wedge \psi\gamma]$ for all $l \to r\ [\psi]$ in $\mathcal{R}$ where l and l' unify with most general unifier γ

Expansion: definition

$$\frac{(\mathcal{E} \uplus \{s \simeq t\ [\varphi]\}, \mathcal{H})}{(\mathcal{E} \cup \mathit{Expd}(s, t, \varphi, p), \mathcal{H} \cup \{s \to t\ [\varphi]\})}$$

if for every γ compatible with φ, $s_{|p}$ reduces and $\mathcal{R} \cup \mathcal{H} \cup \{s \to t\ [\varphi]\}$ is terminating

$\mathit{Expd}(C[l']_p, t, \varphi, p)$ contains equations $C[r\gamma]_p \approx t\gamma\ [\varphi\gamma \wedge \psi\gamma]$ for all $l \to r\ [\psi]$ in $\mathcal{R}$ where l and l' unify with most general unifier γ

Idea: Exhaustive case analysis, generate induction hypothesis.
(Closely related: narrowing.)

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x,y',z') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \wedge\, y' = y+1 \wedge z' = z+y]\}, \mathcal{H})$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$(\mathcal{E} \uplus \{\mathsf{u}(x,y',z') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y]\}, \mathcal{H})$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x, y'+1, z'+y') \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge y' \leq x]\} \\ \quad \cup \{z' \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x, y'+1, z'+y') \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge y' \leq x]\} \\ \quad \cup \{z' \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'+1,z'+y') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge y' \leq x]\} \\ \quad \cup \{z' \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ ,\mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'+1,z'+y') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \land x = x'+1 \\ \qquad \land y' = y+1 \land z' = z+y \land y' \leq x]\} \\ \quad \cup \{z' \approx x + \mathsf{u}(x',y,z)\ [x \geq y \land x = x'+1 \\ \qquad \land y' = y+1 \land z' = z+y \land \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \land x = x'+1 \\ \qquad \land y' = y+1 \land z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'+1,z'+y') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\ y' = y+1 \wedge z' = z+y \wedge y' \leq x]\} \\ \quad \cup\ \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\ y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\ y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x, y'+1, z'+y') \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge y' \leq x]\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x, {\color{red}y''}, {\color{red}z''}) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge\, z' = z + y \wedge y' \leq x {\color{red}\wedge y'' = y' + 1 \wedge z'' = z' + y'}]\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup\, \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ ,\, \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge\, z' = z + y \wedge y' \leq x \wedge y'' = y' + 1 \wedge z'' = z' + y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x, y'', z'') \approx x + \mathsf{u}(x', y', z') \; [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge\, z' = z + y \wedge y' \leq x \wedge y'' = y' + 1 \wedge z'' = z' + y']\} \\ \quad \cup \{z' \approx x + z \; [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z) \; [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z + y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z' + y']\} \\ \quad \cup\, \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup\, \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\ z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\ y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\ y' = y+1 \wedge z' = z+y]\})\ [y := y', y' := y'', z := z', z' := z''] \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\})\ [y := y', y' := y'', z := z', z' := z''] \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup\, \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\})\ [y := y', y' := y'', z := z', z' := z''] \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \qquad \wedge\, z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup\, \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \qquad \wedge\, y' = y+1 \wedge z' = z+y]\}) \ [y := y', y' := y'', z := z', z' := z''] \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{\mathsf{u}(x,y'',z'') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge\, z' = z + y \wedge y' \leq x \wedge y'' = y' + 1 \wedge z'' = z' + y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\})\ [y := y', y' := y'', z := z', z' := z''] \end{array}$$

Expansion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{x + \mathsf{u}(x', y', z') \approx x + \mathsf{u}(x', y', z')\ [x \geq y \wedge x = x' + 1 \wedge y' = \\ \qquad y + 1 \wedge z' = z + y \wedge y' \leq x \wedge y'' = y' + 1 \wedge z'' = z' + y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z)\ [x \geq y \wedge x = x' + 1 \\ \qquad \wedge\, y' = y + 1 \wedge z' = z + y]\}) \end{array}$$

Deletion: definition

$$\frac{(\mathcal{E} \uplus \{s \simeq t\ [\varphi]\}, \mathcal{H})}{(\mathcal{E}, \mathcal{H})}$$

if $s \equiv t$ or φ is unsatisfiable

Idea: Delete trivial inductive theorems.

Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{x + \mathsf{u}(x',y',z') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x'+1 \wedge y' = \\ \quad y+1 \wedge z' = z+y \wedge y' \leq x \wedge y'' = y'+1 \wedge z'' = z'+y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \wedge z' = z+y \\ \quad \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \\ \quad \wedge z' = z+y]\}) \end{array}$$

Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{x + \mathsf{u}(x',y',z') \approx x + \mathsf{u}(x',y',z')\ [x \geq y \wedge x = x' + 1 \wedge y' = \\ \quad y + 1 \wedge z' = z + y \wedge y' \leq x \wedge y'' = y' + 1 \wedge z'' = z' + y']\} \\ \quad \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y \\ \quad \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \quad \wedge z' = z + y]\}) \end{array}$$

Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\ \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\ \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \\ \quad \{z' \approx x + z \; [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y \\ \quad \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z) \; [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \quad \wedge z' = z + y]\}) \end{array}$$

EQ-Deletion: definition

$$\frac{(\mathcal{E} \uplus \{C[s_1, \ldots, s_n] \approx C[t_1, \ldots, t_n] \ [\varphi]\}, \mathcal{H})}{(\mathcal{E} \cup \{C[\vec{s}] \approx C[\vec{t}] \ [\varphi \land \neg \bigwedge_{i=1}^{n} (s_i = t_i)]\}, \mathcal{H})}$$

if $s_1, \ldots, s_n, t_1, \ldots, t_n$ all logical terms

Idea: If all arguments to the same context become equal, we're done.

EQ-Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{z' \approx x + z \ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y \\ \qquad \wedge \neg(y' \leq x)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z) \ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge z' = z + y]\}) \end{array}$$

EQ-Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{z' \approx x + z \; [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y \\ \qquad \wedge \neg(y' \leq x) \wedge \neg(z' = x + z)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x, y', z') \to x + \mathsf{u}(x', y, z) \; [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge z' = z + y]\}) \end{array}$$

EQ-Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$$\begin{array}{l} (\mathcal{E} \cup \{z' \approx x + z\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \wedge z' = z + y \\ \qquad \wedge \neg(y' \leq x) \wedge \neg(z' = x + z)]\} \\ , \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x' + 1 \wedge y' = y + 1 \\ \qquad \wedge z' = z + y]\}) \end{array}$$

EQ-Deletion: example

$$\mathcal{R} = \left\{ \begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array} \right\}$$

$(\mathcal{E}$

$$, \mathcal{H} \cup \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \wedge y' = y+1 \wedge z' = z+y]\})$$

Postulate: definition

$$\frac{(\mathcal{E}, \mathcal{H})}{(\mathcal{E} \uplus \{s \approx t \ [\varphi]\}, \mathcal{H})}$$

Postulate: example

$\mathcal{R}$:

$$
\begin{array}{rcll}
\mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\
\mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\
\mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\
\mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)]
\end{array}
$$

Goal:

$$(\ \{\mathsf{sum}_1(x) \approx \mathsf{sum}_2(x) \ [\top]\}, \emptyset \ )$$

Postulate: example

$\mathcal{R}$:

$$
\begin{array}{lcll}
\mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\
\mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\
\mathsf{sum}_2(x) & \to & \mathsf{u}(x, 0, 0) & \\
\mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\
\mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)]
\end{array}
$$

Goal:

$$
\begin{array}{l}
(\ \{\mathsf{sum}_1(x) \approx \mathsf{sum}_2(x)\ [\top], \\
\quad \mathsf{u}(x, y, z) \approx x + \mathsf{u}(x', y, z)\ [x \geq y \land x = x' + 1 \\
\quad \land\, y' = y + 1 \land z' = z + y]\}, \emptyset\ )
\end{array}
$$

Postulate: example

$\mathcal{R}$:

$$\begin{array}{lcll} \mathsf{sum}_1(x) & \to & 0 & [x \leq 0] \\ \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\ \mathsf{sum}_2(x) & \to & \mathsf{u}(x,0,0) & \\ \mathsf{u}(x,i,z) & \to & \mathsf{u}(x,i+1,z+i) & [i \leq x] \\ \mathsf{u}(x,i,z) & \to & z & [\neg(i \leq x)] \end{array}$$

Goal:

$$\begin{array}{l} (\ \{\mathsf{sum}_1(x) \approx \mathsf{sum}_2(x)\ [\top]\}, \\ \ \ \{\mathsf{u}(x,y',z') \to x + \mathsf{u}(x',y,z)\ [x \geq y \wedge x = x'+1 \\ \ \ \ \wedge\, y' = y+1 \wedge z' = z+y]\}\ ) \end{array}$$

Overview

What Typically Happens

$$
\begin{array}{llcll}
1. & \mathsf{sum}_1(x) & \rightarrow & 0 & [x \leq 0] \\
2. & \mathsf{sum}_1(x) & \rightarrow & x + \mathsf{sum}_1(x-1) & [x > 0] \\
3. & \mathsf{sum}_2(x) & \rightarrow & \mathsf{u}(x, 0, 0) & \\
4. & \mathsf{u}(x, i, z) & \rightarrow & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\
5. & \mathsf{u}(x, i, z) & \rightarrow & z & [\neg(i \leq x)]
\end{array}
$$

Goals:

$\mathsf{sum}_1(x) \approx \mathsf{sum}_2(x)\ [\top]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$

Goals:

$\mathsf{sum}_1(x) \approx \mathsf{u}(x, 0, 0)\ [\top]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$

Goals:

$x + \mathsf{sum}_1(x-1) \approx \mathsf{u}(x, 0, 0)\ [x > 0]$

$0 \approx \mathsf{u}(x, 0, 0)\ [x \leq 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$

Goals:

$x + \mathsf{sum}_1(x-1) \approx \mathsf{u}(x, 0, 0)\ [x > 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$

Goals:

$x + \mathsf{sum}_1(x-1) \approx \mathsf{u}(x, 0+1, 0+0)\ [x > 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$

Goals:

$x + \mathsf{sum}_1(x') \approx \mathsf{u}(x, 1, 0)\ [x > 0 \wedge x' = x - 1]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$

Goals:

$x + \mathsf{u}(x', 0, 0) \approx \mathsf{u}(x, 1, 0)\ [x > 0 \wedge x' = x - 1]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x,0,0)$	
4.	$\mathsf{u}(x,i,z)$	$\to$	$\mathsf{u}(x,i+1,z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x,i,z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x,0,0)$	$[\top]$
H2	$\mathsf{u}(x,1,0)$	$\to$	$x + \mathsf{u}(x',0,0)$	$[x > 0 \wedge x' = x - 1]$

Goals:

$x + \mathsf{u}(x',0,0) \approx \mathsf{u}(x,1+1,0+1)\ [x > 0 \wedge x' = x - 1 \wedge x' > 0]$

$x + \mathsf{u}(x',0,0) \approx 0\ [x > 0 \wedge x' = x - 1 \wedge x' \leq 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x,0,0)$	
4.	$\mathsf{u}(x,i,z)$	$\to$	$\mathsf{u}(x,i+1,z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x,i,z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x,0,0)$	$[\top]$
H2	$\mathsf{u}(x,1,0)$	$\to$	$x + \mathsf{u}(x',0,0)$	$[x > 0 \wedge x' = x - 1]$

Goals:

$x + \mathsf{u}(x',0,0) \approx \mathsf{u}(x,1+1,0+1)\ [x > 0 \wedge x' = x - 1 \wedge x' > 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$
H2	$\mathsf{u}(x, 1, 0)$	$\to$	$x + \mathsf{u}(x', 0, 0)$	$[x > 0 \land x' = x - 1]$

Goals:

$$x + \mathsf{u}(x', 0+1, 0+0) \approx \mathsf{u}(x, 1+1, 0+1)\ [x > 0 \land x' = x-1 \land x' > 0]$$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$
H2	$\mathsf{u}(x, 1, 0)$	$\to$	$x + \mathsf{u}(x', 0, 0)$	$[x > 0 \land x' = x - 1]$

Goals:

$x + \mathsf{u}(x', 1, 0) \approx \mathsf{u}(x, 2, 1)\ [x > 0 \land x' = x - 1 \land x' > 0]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \le 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \le x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \le x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$
H2	$\mathsf{u}(x, 1, 0)$	$\to$	$x + \mathsf{u}(x', 0, 0)$	$[x > 0 \land x' = x - 1]$
H3	$\mathsf{u}(x, 2, 1)$	$\to$	$x + \mathsf{u}(x', 1, 0)$	$[x \ge 1 \land x' = x - 1]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	$[\top]$
H2	$\mathsf{u}(x, 1, 0)$	$\to$	$x + \mathsf{u}(x', 0, 0)$	$[x > 0 \wedge x' = x - 1]$
H3	$\mathsf{u}(x, 2, 1)$	$\to$	$x + \mathsf{u}(x', 1, 0)$	$[x \geq 1 \wedge x' = x - 1]$
H4	$\mathsf{u}(x, 3, 3)$	$\to$	$x + \mathsf{u}(x', 2, 1)$	$[x \geq 2 \wedge x' = x - 1]$

What Typically Happens

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x,0,0)$	
4.	$\mathsf{u}(x,i,z)$	$\to$	$\mathsf{u}(x,i+1,z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x,i,z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x,0,0)$	$[\top]$
H2	$\mathsf{u}(x,1,0)$	$\to$	$x + \mathsf{u}(x',0,0)$	$[x > 0 \wedge x' = x-1]$
H3	$\mathsf{u}(x,2,1)$	$\to$	$x + \mathsf{u}(x',1,0)$	$[x \geq 1 \wedge x' = x-1]$
H4	$\mathsf{u}(x,3,3)$	$\to$	$x + \mathsf{u}(x',2,1)$	$[x \geq 2 \wedge x' = x-1]$
H5	$\mathsf{u}(x,4,6)$	$\to$	$x + \mathsf{u}(x',3,3)$	$[x \geq 3 \wedge x' = x-1]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	0	$[x \leq 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, 0, 0)$	
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$

Goals:

$\mathsf{sum}_1(x) \approx \mathsf{sum}_2(x)\ [\top]$

Use Different Notation!

$$
\begin{array}{llcll}
1. & \mathsf{sum}_1(x) & \to & c0 & [x \leq 0 \wedge c0 = 0] \\
2. & \mathsf{sum}_1(x) & \to & x + \mathsf{sum}_1(x-1) & [x > 0] \\
3. & \mathsf{sum}_2(x) & \to & \mathsf{u}(x, c1, c2) & [c1 = 0 \wedge c2 = 0] \\
4. & \mathsf{u}(x, i, z) & \to & \mathsf{u}(x, i+1, z+i) & [i \leq x] \\
5. & \mathsf{u}(x, i, z) & \to & z & [\neg(i \leq x)]
\end{array}
$$

Goals:

$\mathsf{sum}_1(x) \approx \mathsf{u}(x, c1, c2)\ [c1 = 0 \wedge c2 = 0]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{sum}_1(x - 1) \approx \mathsf{u}(x, c1, c2)\ [c1 = 0 \wedge c2 = 0 \wedge x > 0]$

$c0 \approx \mathsf{u}(x, c1, c2)\ [c1 = 0 \wedge c2 = 0 \wedge x \leq 0 \wedge c0 = 0]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{sum}_1(x-1) \approx \mathsf{u}(x, c1, c2)\ [c1 = 0 \wedge c2 = 0 \wedge x > 0]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{sum}_1(x-1) \approx \mathsf{u}(x, c1+1, c2+c1)\ [c1 = 0 \wedge c2 = 0 \wedge x > 0]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{sum}_1(x') \approx \mathsf{u}(x, i, z)\ [c1 = 0 \wedge c2 = 0 \wedge x > 0 \wedge x' = x - 1 \wedge i = c1 + 1 \wedge z = c1 + c2]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{u}(x', c1, c2) \approx \mathsf{u}(x, i, z)$ $[c1 = 0 \wedge c2 = 0 \wedge x > 0 \wedge x' = x - 1 \wedge i = c1 + 1 \wedge z = c1 + c2]$

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x - 1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i + 1, z + i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{u}(x', c1, c2) \approx \mathsf{u}(x, i, z)\ [c1 = 0 \wedge c2 = 0 \wedge x > 0 \wedge x' = x - 1 \wedge i = c1 + 1 \wedge z = c1 + c2]$

Generalisation: Drop initialisations

Use Different Notation!

1.	$\mathsf{sum}_1(x)$	$\to$	$c0$	$[x \leq 0 \wedge c0 = 0]$
2.	$\mathsf{sum}_1(x)$	$\to$	$x + \mathsf{sum}_1(x-1)$	$[x > 0]$
3.	$\mathsf{sum}_2(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$
4.	$\mathsf{u}(x, i, z)$	$\to$	$\mathsf{u}(x, i+1, z+i)$	$[i \leq x]$
5.	$\mathsf{u}(x, i, z)$	$\to$	z	$[\neg(i \leq x)]$
H1	$\mathsf{sum}_1(x)$	$\to$	$\mathsf{u}(x, c1, c2)$	$[c1 = 0 \wedge c2 = 0]$

Goals:

$x + \mathsf{u}(x', c1, c2) \approx \mathsf{u}(x, i, z)\ [x > 0 \wedge x' = x - 1 \wedge i = c1 + 1 \wedge z = c1 + c2]$

Generalisation: Drop initialisations

Overview

Shoulders of Giants

Shoulders of Giants

- various kinds of constrained rewriting

Shoulders of Giants

- various kinds of constrained rewriting
  (But: most were fundamentally limited to the integers)

Shoulders of Giants

- various kinds of constrained rewriting
  (But: most were fundamentally limited to the integers)
- long history of unconstrained rewriting induction, e.g. [Reddy 1990]

Shoulders of Giants

- various kinds of constrained rewriting
  (But: most were fundamentally limited to the integers)
- long history of unconstrained rewriting induction, e.g. [Reddy 1990]
  (But: lemma generation methods do not obviously extend)

Shoulders of Giants

- various kinds of constrained rewriting
  (But: most were fundamentally limited to the integers)
- long history of unconstrained rewriting induction, e.g. [Reddy 1990]
  (But: lemma generation methods do not obviously extend)
- rewriting induction for a form of constrained rewriting

Shoulders of Giants

- various kinds of constrained rewriting
  (But: most were fundamentally limited to the integers)
- long history of unconstrained rewriting induction, e.g. [Reddy 1990]
  (But: lemma generation methods do not obviously extend)
- rewriting induction for a form of constrained rewriting
  (But: only very complex and relatively weak lemma generation)

Implementation and Experiments

Implementation and Experiments

- C2LCTRS: automatic tool to translate C programs to LCTRSs
  http://www.trs.cm.is.nagoya-u.ac.jp/c2lctrs/

Implementation and Experiments

- C2LCTRS: automatic tool to translate C programs to LCTRSs
  http://www.trs.cm.is.nagoya-u.ac.jp/c2lctrs/
- Ctrl: automatic tool to prove equivalence of LCTRS functions
  http://cl-informatik.uibk.ac.at/software/ctrl/

Implementation and Experiments

- C2LCTRS: automatic tool to translate C programs to LCTRSs
  http://www.trs.cm.is.nagoya-u.ac.jp/c2lctrs/
- Ctrl: automatic tool to prove equivalence of LCTRS functions
  http://cl-informatik.uibk.ac.at/software/ctrl/

function	YES	NO	MAYBE	time
sum	9	0	4	2.4
fib	4	6	3	6.6
sumfrom	3	1	2	1.9
strlen	1	0	5	7.2
strcpy	3	0	3	11.5
arrsum	1	0	0	4.2
fact	1	0	0	2.4
total	22	7	17	5.9

Experiments with student code

Implementation and Experiments

- C2LCTRS: automatic tool to translate C programs to LCTRSs
  http://www.trs.cm.is.nagoya-u.ac.jp/c2lctrs/
- Ctrl: automatic tool to prove equivalence of LCTRS functions
  http://cl-informatik.uibk.ac.at/software/ctrl/

function	YES	NO	MAYBE	time
sum	9	0	4	2.2
fib	10	1	2	5.9
sumfrom	3	0	3	2.3
strlen	2	0	4	6.0
strcpy	5	0	1	14.1
arrsum	1	0	0	4.2
fact	1	0	0	2.5
total	31	1	14	5.9

Experiments with student code and adapted teacher code

Conclusion

- Logically Constrained Term Rewrite Systems for automated reasoning and program analysis

Conclusion

- Logically Constrained Term Rewrite Systems
  for automated reasoning and program analysis
- Equivalence-preserving frontend to translate
  from C fragment on integers, arrays, ... to LCTRSs

Conclusion

- Logically Constrained Term Rewrite Systems for automated reasoning and program analysis
- Equivalence-preserving frontend to translate from C fragment on integers, arrays, ... to LCTRSs
- Constrained rewriting induction to prove equivalence of functions defined by LCTRSs

Conclusion

- Logically Constrained Term Rewrite Systems for automated reasoning and program analysis
- Equivalence-preserving frontend to translate from C fragment on integers, arrays, ... to LCTRSs
- Constrained rewriting induction to prove equivalence of functions defined by LCTRSs
- Lemma generation technique suited to problems from imperative programs

Conclusion

- Logically Constrained Term Rewrite Systems for automated reasoning and program analysis
- Equivalence-preserving frontend to translate from C fragment on integers, arrays, ... to LCTRSs
- Constrained rewriting induction to prove equivalence of functions defined by LCTRSs
- Lemma generation technique suited to problems from imperative programs
- Conference paper at APLAS 2014

Conclusion

- Logically Constrained Term Rewrite Systems
  for automated reasoning and program analysis
- Equivalence-preserving frontend to translate
  from C fragment on integers, arrays, ... to LCTRSs
- Constrained rewriting induction to prove equivalence of functions defined by LCTRSs
- Lemma generation technique suited to problems from imperative programs
- Conference paper at APLAS 2014
- Full version: `http://arxiv.org/abs/1409.0166`

Conclusion

- Logically Constrained Term Rewrite Systems for automated reasoning and program analysis
- Equivalence-preserving frontend to translate from C fragment on integers, arrays, ... to LCTRSs
- Constrained rewriting induction to prove equivalence of functions defined by LCTRSs
- Lemma generation technique suited to problems from imperative programs
- Conference paper at APLAS 2014
- Full version: http://arxiv.org/abs/1409.0166

Questions?