Symbolic Computation Support for Complexity Analysis and the PURRS Project

Roberto BAGNARA, Alessandro ZACCAGNINI,
Enea ZAFFANELLA, and Tatiana ZOLO
Department of Mathematics
University of Parma, Italy

http://www.cs.unipr.it/purrs/

Plan of the Talk

① Complexity Analysis and Symbolic Computation

② Classes of Recurrence Relations

③ Solving and Approximating Recurrence Relations

④ Symbolic Manipulation of Solutions and Approximations

⑤ The PURRS Library

Tracking the Usage of Resources

What?

➜ We are interested in those properties of complex systems that deal with resource usage:

- computation time,
- required memory space,
- network bandwidth used, . . .

Why?

➜ verifying that the deadlines of hard real-time systems are met;

➜ deciding whether a mobile agent should be allowed to run in a certain context;

➜ guiding the application of optimizing program transformations;

➜ assisting the programmer in reasoning on programs:
$\Longrightarrow$ particularly useful with high level languages where introducing efficiency bugs is very easy.

Automatic Complexity Analysis: The Domain of Discourse

- $\wp(\mathbb{R}_\infty^{\mathbb{N}})$: (possibly infinite) sets of (infinite) sequences of real numbers.
- A sequence expresses the cost of one process in terms of some input measure:
  - cost may be in terms of clock cycles, number of statements executed, memory used, number of packets exchanged over the network, . . . ;
  - a process may be a piece of software but also a communication protocol;
  - the input measure can be any metric of the input of a program/procedure, or, say, the number of participants to some synchronization protocol.
- We have sets of sequences to capture approximation.

The Need for Powerful Symbolic Computation

Elements of $\wp(\mathbb{R}_\infty^\mathbb{N})$ are generated by:

- ➜ imposing a recurrence relation that a sequence must satisfy to capture recursion;
- ➜ computing additions in order to approximate sequential composition;
- ➜ computing approximations of set union in order to capture conditionals;
- ➜ ...

The Need for Powerful Symbolic Computation

Elements of $\wp(\mathbb{R}_\infty^{\mathbb{N}})$ are generated by:

- imposing a recurrence relation that a sequence must satisfy to capture recursion;
- computing additions in order to approximate sequential composition;
- computing approximations of set union in order to capture conditionals;
- ...

Approximations based on lower and upper bounds

- Chosen a class of *boundary functions* $\mathfrak{B} \in \wp(\mathbb{R}_\infty^{\mathbb{N}})$,
- we may represent the subset of $\wp(\mathbb{R}_\infty^{\mathbb{N}})$ defined by

$$\mathfrak{F} \stackrel{\text{def}}{=} \Big\{ F \in \wp(\mathbb{R}_\infty^{\mathbb{N}}) \;\Big|\; \exists l, u \in \mathfrak{B} \,.\, \forall f \in F : l \le f \le u \Big\}.$$

- Given $b_1, b_2 \in \mathfrak{B}$, we need to approximate, within $\mathfrak{B}$, $\max\{b_1, b_2\}$ from above and $\min\{b_1, b_2\}$ from below.

Classes of Recurrence Relations

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

Classes of Recurrence Relations

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

➜ Linear recurrences of finite order with constant coefficients

$x_n = 5x_{n-1} - 6x_{n-2} + n^2$

Classes of Recurrence Relations

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

- ➜ Linear recurrences of finite order with constant coefficients
  $x_n = 5x_{n-1} - 6x_{n-2} + n^2$
- ➜ Linear recurrences of finite order with variable coefficients
  $x_n = \frac{1}{n} x_{n-1} + 2$

Classes of Recurrence Relations

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

- Linear recurrences of finite order with constant coefficients

  $x_n = 5x_{n-1} - 6x_{n-2} + n^2$
- Linear recurrences of finite order with variable coefficients

  $x_n = \frac{1}{n}x_{n-1} + 2$
- Linear recurrences of infinite order

  $x_n = n + \frac{2}{n}\sum_{k=0}^{n-1} x_k$

CLASSES OF RECURRENCE RELATIONS

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

- ➜ Linear recurrences of finite order with constant coefficients

  $x_n = 5x_{n-1} - 6x_{n-2} + n^2$

- ➜ Linear recurrences of finite order with variable coefficients

  $x_n = \frac{1}{n} x_{n-1} + 2$

- ➜ Linear recurrences of infinite order

  $x_n = n + \frac{2}{n} \sum_{k=0}^{n-1} x_k$

- ➜ Non-linear recurrences of finite order

  $x_n = 3x_{n-1}^2$

Classes of Recurrence Relations

For each class of recurrence relations a specific approach is chosen for the computation or approximation of the solutions.

- Linear recurrences of finite order with constant coefficients
  $x_n = 5x_{n-1} - 6x_{n-2} + n^2$
- Linear recurrences of finite order with variable coefficients
  $x_n = \frac{1}{n} x_{n-1} + 2$
- Linear recurrences of infinite order
  $x_n = n + \frac{2}{n} \sum_{k=0}^{n-1} x_k$
- Non-linear recurrences of finite order
  $x_n = 3x_{n-1}^2$
- Divide-et-impera recurrences
  $x_n = 2x_{n/2} + n - 1$

Linear Rec. of Finite Order with Constant Coeff.

A general solution method is available, based on the characteristic equation of the recurrence:

$$\lambda^k = a_1\lambda^{k-1} + \cdots + a_{k-1}\lambda + a_k.$$

➜ Roots of equation $+$ symbolic summations $\implies$ exact solution.

Linear Rec. of Finite Order with Constant Coeff.

A general solution method is available, based on the characteristic equation of the recurrence:

$$\lambda^k = a_1 \lambda^{k-1} + \cdots + a_{k-1}\lambda + a_k.$$

- Roots of equation $+$ symbolic summations $\Longrightarrow$ exact solution.
- Finding the roots is feasible in many cases:
  - order-reduction transformation;
  - square-free factorization;
  - identification of “small” rational roots;
  - direct algebraic solution (up to 4th order);
  - other factorization methods ...
- Computing symbolic summations is also feasible in many cases:
  - linear combinations of polynomials and exponential, their products;
  - other special classes of functions.

Linear Rec. of Finite Order with Constant Coeff.

A general solution method is available, based on the characteristic equation of the recurrence:

$$\lambda^k = a_1\lambda^{k-1} + \cdots + a_{k-1}\lambda + a_k.$$

- ➜ Roots of equation + symbolic summations $\implies$ exact solution.
- ➜ Finding the roots is feasible in many cases:
  - ➜ order-reduction transformation;
  - ➜ square-free factorization;
  - ➜ identification of “small” rational roots;
  - ➜ direct algebraic solution (up to 4th order);
  - ➜ other factorization methods . . .
- ➜ Computing symbolic summations is also feasible in many cases:
  - ➜ linear combinations of polynomials and exponential, their products;
  - ➜ other special classes of functions.
- ➜ Approximating the roots and the summations in the remaining cases.

Linear Rec. of Finite Order with Variable Coeff.

No general solution method is known.

Linear Rec. of Finite Order with Variable Coeff.

No general solution method is known.

- ➜ A solution method is available for the 1st order case (possibly after the order-reduction step). Let $\Pi(n) = \prod_{k=1}^{n} \alpha(k)$:

$$
\begin{tikzcd}
x_n = \alpha(n) x_{n-1} + p(n) \arrow[r, "x_n = \Pi(n) y_n"] \arrow[d, dotted] & y_n = y_{n-1} + \frac{p(n)}{\Pi(n)} \arrow[d, "\text{solve}"] \\
x_n = \Pi(n)\left(x_0 + \sum_{k=1}^{n} \frac{p(k)}{\Pi(k)}\right) & y_n = y_0 + \sum_{k=1}^{n} \frac{p(k)}{\Pi(k)} \arrow[l, "y_n = \frac{x_n}{\Pi(n)}"']
\end{tikzcd}
$$

LINEAR REC. OF FINITE ORDER WITH VARIABLE COEFF.

No general solution method is known.

➜ A solution method is available for the 1st order case (possibly after the order-reduction step). Let $\Pi(n) = \prod_{k=1}^{n} \alpha(k)$:

$$\begin{array}{ccc}
x_n = \alpha(n)x_{n-1} + p(n) & \xrightarrow{x_n = \Pi(n) y_n} & y_n = y_{n-1} + \frac{p(n)}{\Pi(n)} \\
\Big\downarrow & & \Big\downarrow \text{solve} \\
x_n = \Pi(n)\Big(x_0 + \sum_{k=1}^{n} \frac{p(k)}{\Pi(k)}\Big) & \xleftarrow{y_n = \frac{x_n}{\Pi(n)}} & y_n = y_0 + \sum_{k=1}^{n} \frac{p(k)}{\Pi(k)}
\end{array}$$

➜ Other methods (e.g., Zeilberger's algorithm) can be applied to find polynomial and hypergeometric solutions for higher-order recurrences.

The analysis of the average case for quicksort leads to

$$x_n = n + \frac{2}{n}\sum_{k=0}^{n-1} x_k.$$

Linear Recurrences of Infinite Order

The analysis of the average case for quicksort leads to

$$x_n = n + \frac{2}{n} \sum_{k=0}^{n-1} x_k .$$

➜ Here x_n depends on *all* previous values, and not only on a fixed number of them.

LINEAR RECURRENCES OF INFINITE ORDER

The analysis of the average case for quicksort leads to

$$x_n = n + \frac{2}{n} \sum_{k=0}^{n-1} x_k.$$

- ➜ Here x_n depends on *all* previous values, and not only on a fixed number of them.
- ➜ All values have the same weight $2/n$: therefore x also satisfies

  $$x_n = \left(1 + \frac{1}{n}\right) x_{n-1} + 2 - \frac{1}{n},$$

  which is a 1st order linear recurrence with variable coefficients.

Linear Recurrences of Infinite Order

The analysis of the average case for quicksort leads to

$$x_n = n + \frac{2}{n} \sum_{k=0}^{n-1} x_k.$$

- Here x_n depends on *all* previous values, and not only on a fixed number of them.
- All values have the same weight $2/n$: therefore x also satisfies

  $$x_n = \left(1 + \frac{1}{n}\right) x_{n-1} + 2 - \frac{1}{n},$$

  which is a 1st order linear recurrence with variable coefficients.
- By applying the solution method above plus approximations, we obtain

  $$x_n = 2(n+1)\log n + n\big(x_0 - 3 + 2\gamma\big) + \epsilon(n),$$

  where γ is Euler's constant and $\epsilon(n) \in \mathcal{O}(1)$.

NON-LINEAR RECURRENCES OF FINITE ORDER

No general solution method is known.

Non-Linear Recurrences of Finite Order

No general solution method is known.

- ➜ Handling special cases:
  - ➜ by means of so-called range transformation, some recurrences can be linearized and previous methods become applicable.

$$
\begin{tikzcd}
x_n = 3x_{n-1}^2 \arrow[r, "\log"] \arrow[d, dotted] & y_n = 2y_{n-1} + \log_2 3 \arrow[d, "\text{solve}"] \\
x_n = 3^{2^n - 1} x_0^{2^n} & y_n = 2^n y_0 + (2^n - 1)\log_2 3 \arrow[l, "\exp"']
\end{tikzcd}
$$

DIVIDE-ET-IMPERA RECURRENCES

- Closed-form solutions may not exist $\implies$ must use approximations.
  - Upper and lower bounds: valid for each $n \in \mathbb{N}$ for which the recurrence is well-defined.

DIVIDE-ET-IMPERA RECURRENCES

- Closed-form solutions may not exist $\implies$ must use approximations.
  - Upper and lower bounds: valid for each $n \in \mathbb{N}$ for which the recurrence is well-defined.
- Recurrences of rank 1

$$x(n) = \alpha x\left(\frac{n}{\beta}\right) + g(n),$$

where $\alpha > 0$, $\beta > 1$ and $g(n)$ is a non-negative, non-decreasing function.

  - Special cases for g: less generality but more efficiency for combinations of polynomials, exponentials, logarithms and factorials.
- Recurrences of higher rank

$$x(n) = \alpha_1 x\left(\frac{n}{\beta_1}\right) + \alpha_2 x\left(\frac{n}{\beta_2}\right) + g(n)$$

  - Other techniques may be used in some cases.
  - Otherwise we may resort to further approximations.

Divide-et-Impera Recurrences

Results in

R. Bagnara, A. Zaccagnini, E. Zaffanella, and T. Zolo, 2003.
The Automatic Solution of Recurrence Relations: "Divide et Impera" Recurrences.

establish very precise lower and upper bounds.

➜ Example (Strassen's algorithm): $x(n) = 7x(n/2) + 18n^2$

$$\frac{24}{7} n^{\frac{\log 7}{\log 2}} - 24n^2 \le x(n) \le \frac{351}{5} n^{\frac{\log 7}{\log 2}} - 24n^2 + \frac{144}{5} n - 3.$$

DIVIDE-ET-IMPERA RECURRENCES

Results in

R. Bagnara, A. Zaccagnini, E. Zaffanella, and T. Zolo, 2003.
The Automatic Solution of Recurrence Relations: "Divide et Impera" Recurrences.

establish very precise lower and upper bounds.

- ➜ Example (Strassen's algorithm): $x(n) = 7x(n/2) + 18n^2$

$$\frac{24}{7} n^{\frac{\log 7}{\log 2}} - 24n^2 \leq x(n) \leq \frac{351}{5} n^{\frac{\log 7}{\log 2}} - 24n^2 + \frac{144}{5} n - 3.$$

- ➜ Example (mergesort algorithm): $x(n) = 2x(n/2) + n - 1$

$$h(n) - 2n + 1 \leq x(n) \leq h(n) + \frac{1}{2} n x(1),$$

where $h(n) = \frac{(n-1)\log n}{\log 2} + \frac{1}{2} n x(1) + 1.$

Divide-et-Impera Recurrences

Results in

R. Bagnara, A. Zaccagnini, E. Zaffanella, and T. Zolo, 2003.
The Automatic Solution of Recurrence Relations: "Divide et Impera" Recurrences.

establish very precise lower and upper bounds.

- ➜ Example (Strassen's algorithm): $x(n) = 7x(n/2) + 18n^2$

$$\frac{24}{7} n^{\frac{\log 7}{\log 2}} - 24n^2 \leq x(n) \leq \frac{351}{5} n^{\frac{\log 7}{\log 2}} - 24n^2 + \frac{144}{5} n - 3.$$

- ➜ Example (mergesort algorithm): $x(n) = 2x(n/2) + n - 1$

$$h(n) - 2n + 1 \leq x(n) \leq h(n) + \frac{1}{2} n x(1),$$

where $h(n) = \frac{(n-1)\log n}{\log 2} + \frac{1}{2} n x(1) + 1$.
$\Longrightarrow$ We thus determined the asymptotic formula $x(n) \sim \frac{n \log n}{\log 2}$.

Manipulation of Infinite Sequences (I)

Do we have $\forall n \in \mathbb{N} : f(n) = 0$?

➜ This is needed in order to verify the solver itself.

MANIPULATION OF INFINITE SEQUENCES (I)

Do we have $\forall n \in \mathbb{N} : f(n) = 0$?

➜ This is needed in order to verify the solver itself.

Find *simple* $u, l\colon \mathbb{N} \to \mathbb{R}_\infty$ approximating $f\colon \mathbb{N} \to \mathbb{R}_\infty$ from above and from below.

➜ This is needed because exact solutions may be too complex.

Manipulation of Infinite Sequences (I)

Do we have $\forall n \in \mathbb{N} : f(n) = 0$?

➜ This is needed in order to verify the solver itself.

Find *simple* $u, l \colon \mathbb{N} \to \mathbb{R}_\infty$ approximating $f \colon \mathbb{N} \to \mathbb{R}_\infty$ from above and from below.

➜ This is needed because exact solutions may be too complex.

Find approximations of $\max\{b_1, b_2\}$ and $\min\{b_1, b_2\}$ from below.

➜ This is needed to compute upper bounds on $\mathfrak{F}$.

Manipulation of Infinite Sequences (I)

Do we have $\forall n \in \mathbb{N} : f(n) = 0$?

➜ This is needed in order to verify the solver itself.

Find *simple* $u, l \colon \mathbb{N} \to \mathbb{R}_\infty$ approximating $f \colon \mathbb{N} \to \mathbb{R}_\infty$ from above and from below.

➜ This is needed because exact solutions may be too complex.

Find approximations of $\max\{b_1, b_2\}$ and $\min\{b_1, b_2\}$ from below.

➜ This is needed to compute upper bounds on $\mathfrak{F}$.

Do we have $\forall n \in \mathbb{N} : f(n) \leq g(n)$?

➜ Useful to provide better lower and upper bounds.

➜ Useful to check that they are indeed lower and upper bounds.

➜ Useful for the applications: e.g., for an optimizing program transformer to automatically decide whether a candidate transformation resulted into an actual improvement.

Manipulation of Infinite Sequences (II)

- Already obtained some results about these problems:

  R. Bagnara and A. Zaccagnini, 2003.

  Checking and Bounding the Solutions of Some Recurrence Relations.

  - The problems are reduced to testing a finite (and usually very small) set of conditions;
  - in several cases these conditions are simple comparisons between integers.
- The proposed techniques share some aspects:
  - identification of dominant terms;
  - exploitation of the properties of restricted classes of functions (polynomials times exponentials, sums of these functions, factorials);
  - divide-et-impera and tests by inductions: break down expressions and evaluate the pieces at a small number of consecutive integers.
- Upper bound computed by replacing terms of the form $an^k\lambda^n$ with $|a|n^k\Lambda^n$, where Λ is an upper bound for the value of $|\lambda|$.

Example: Check the Formula for the Fibonacci Numbers

$$\begin{cases} x_n = x_{n-1} + x_{n-2}, \\ x_0 = 0, \\ x_1 = 1, \end{cases} \qquad \Longrightarrow \qquad x_n = \frac{\lambda_1^n - \lambda_2^n}{\lambda_1 - \lambda_2}, \quad \text{with} \begin{cases} \lambda_1 = \frac{1}{2}(1+\sqrt{5}), \\ \lambda_2 = \frac{1}{2}(1-\sqrt{5}). \end{cases}$$

- ➜ Compute x_0 and x_1 by means of the formula, and check that they agree with data.
- ➜ Compute $x_n - x_{n-1} - x_{n-2}$ and check that it is 0 for $n \geq 2$.
- ➜ We have to verify that $\lambda_1^n - \lambda_1^{n-1} - \lambda_1^{n-2} - \lambda_2^n + \lambda_2^{n-1} + \lambda_2^{n-2} = 0$ for all integers $n \geq 2$.

  $\Longrightarrow$ It can be proved that if this happens for any 6 *consecutive* integers, then it is true for all integers.

PURRS: A Powerful Recurrence Relation Solver

- ➜ A system implementing these ideas being developed at the University of Parma, Italy.
- ➜ Written in C++, but easily interfaceable with other programming languages.

PURRS: A Powerful Recurrence Relation Solver

- A system implementing these ideas being developed at the University of Parma, Italy.
- Written in C++, but easily interfaceable with other programming languages.

Main components:

- a package providing basic computer algebra services;
- several rewriting systems providing specialized simplifications;
- an algebraic equation solver;
- modules to compute closed formulas for symbolic summations (with different efficiency/power ratios);
- modules implementing verification and comparisons.

USING THE PURRS TEST DRIVER

➜ Asking for exact solutions:

```
$ rrs_driver -E -R "2*x(n-1) + 2^(n-1) + n + 1"
x(n) = -3+3*2^n+1/2*2^n*n+2^n*x(0)-n.
$ rrs_driver -E -R "2*x(n-1) + 2^(n-1) + n + 1" -I "x(0) = 1"
x(n) = -3+4*2^n+1/2*2^n*n-n.
$ rrs_driver -E -R "2 ^ x(n-1) + x(n-2)" -I "x(0) = 1"
exact(too_complex).
```

➜ Asking for upper and lower bounds:

```
$ rrs_driver -UL -R "2*x(n/2) + log(n)"
x(n) >= log(2)*n+1/2*x(1)*n-log(2)-log(n).
x(n) =< 3*log(2)*n+x(1)*n-2*log(2)+log(n)*n-log(n).
$ rrs_driver -UL -R "2*x(n/2) + log(n)" -I"x(1) = 2*p + q"
x(n) >= log(2)*n-log(2)+1/2*q*n-log(n)+p*n.
x(n) =< 3*log(2)*n-2*log(2)+q*n+log(n)*n-log(n)+2*p*n.
```

Conclusion

- ➜ Research and implementation work is ongoing.
- ➜ The aim is to provide complete symbolic computation support for fully automatic complexity analysis.
  $\Longrightarrow$ Even going beyond the language of (generalized) recurrences:

$$x_n = \max_{0 \le k \le n-1} (x_{n-1-k} + x_k) + 2n.$$

- ➜ Collaborations have been started
  - ➜ University of Réunion (France);
  - ➜ University of Leeds (U.K.);
  - ➜ UPM?
- ➜ A demo of the recurrence relation solver is online at `http://www.cs.unipr.it/purrs/`.