Proving Correctness of Imperative Programs by Linearizing Constrained Horn Clauses

Emanuele De Angelis (University “d’Annunzio”, Pescara, Italy)
Fabio Fioravanti (University “d’Annunzio”, Pescara, Italy)
Alberto Pettorossi (University “Tor Vergata”, Rome, Italy)
Maurizio Proietti (IASI-CNR, Rome, Italy)

Program Correctness

- Imperative, sequential programs (integer variables, assignments, conditionals, while, jumps)
- Partial correctness specification (Hoare triple):

  $$\{\varphi\}\ \mathit{prog}\ \{\psi\}$$

  where the assertions φ and ψ are first order formulas.
- If the initial values of the program variables satisfy the precondition φ and *prog* terminates, then the final values of the program variables satisfy the postcondition ψ.
- Problem: Automatically verify the validity of $\{\varphi\}\ \mathit{prog}\ \{\psi\}$.

Translating Partial Correctness to Constrained Horn Clauses

- Suppose φ and ψ are linear arithmetic constraints (LA constraints).
- Partial correctness specifications

  {n≥1} x=0; y=0; while (x<n) {x=x+1; y=y+2} {y>x}

  can be translated into Horn clauses with LA constraints (aka Constrained Horn Clauses, or CHCs):

```
          p(X, Y, N) :- N≥1, X=0, Y=0.                    %Initialization
          p(X1, Y1, N) :- X<N, X1=X+1, Y1=Y+2, p(X, Y, N).  %Loop
Goal:     false :- Y≤X, X≥ N, p(X, Y, N).                 %Loop exit
```

- The program is partially correct if the set of clauses is satisfiable, i.e., it has an **LA**-model (a model that interprets LA constraints as relations on the integers).
- **p(X, Y, N)** denotes a loop invariant (in implicit form).

Satisfiability and LA-Solvability

- Problem: find loop invariants (and, in general, assertions that prove partial correctness) in closed form.

- An LA-solution of a set S of CHCs is a mapping

  Σ: Atom $\rightarrow$ $\mathbf{C_{LA}}$

  such that, for every clause A_0 :- c, A_1,…,A_n in S,

  LA $\models \forall(c, \Sigma(A_1),\ldots,\Sigma(A_n) \rightarrow \Sigma(A_0))$

- A set of CHCs is LA-solvable if it has an LA-solution [Bjørner-McMillan-Rybalchenko 12].

- An LA-solution (if it exists) is an LA overapproximation of the least **LA**-model.

… Satisfiability and LA-Solvability

- **p(X, Y, N) ↦ (X=0, Y=0, N≥1) ∨ Y>X** is an LA-solution of

  p(X, Y, N) :- N≥1, X=0, Y=0.
  p(X1, Y1, N) :- X<N, X1=X+1, Y1=Y+2, p(X, Y, N).
  false :- Y≤X, X≥N, p(X, Y, N).

- **(X=0, Y=0, N≥1) ∨ Y>X** is a loop invariant in closed form.

- LA-solvability implies satisfiability, but not vice versa. There are **LA**-models not expressible as (finite) LA constraints.

- Satisfiability is undecidable and not semidecidable. LA-solvability is semidecidable. (LA constraints are r.e. and entailment is decidable.)

LA-Solvers

- Effective solvers for checking LA-solvability of constrained Horn clauses are available:

  QARMC [Rybalchenko et al.],
  Z3 [deMoura-Bjørner],
  Eldarica [Rümmer et al.]
  MathSAT [Griggio et al.],
  SeaHorn [Gurfinkel,Navas],
  VeriMAP [DeAngelis et al.],
  CPA [Gallagher-Kafle],
  TRACER [Jaffar et al.]

- CLP systems focus on proving unsatisfiability and do not compute an answer constraint when the program and the goal are satisfiable (and often they do not terminate in this case)

Limitations of Linear Arithmetic Assertions

- Not very expressive.
- Example: computing Fibonacci numbers

fibonacci: **while (n>0) {**
t=u;
u=u+v;
v=t;
n=n-1 }

{n=N, N>=0, u=1, v=0, t=0} *fibonacci* {fib(N,u)}

- The postcondition of *fibonacci* cannot be specified by using linear arithmetic constraints *only*.

Our Contributions

- Extension of partial correctness specifications where preconditions and postconditions are predicates defined by any set of CHCs (in particular, the **fib** predicate can be defined).
- Translation of extended partial correctness specifications into CHCs.
- A limitation of LA-solving: The CHCs derived by translating some partial correctness specifications, e.g., *fibonacci*, are satisfiable but not LA-solvable.
- Linearization: A transformation that increases the power of LA-solving (and transforms the CHCs for *fibonacci* into LA-solvable clauses).

Horn Clause Specifications

- Functional Horn specifications:
  {z1= P1, . . . , zs= Ps, **pre**(P1, . . . , Ps)} *prog* {**f**(P1, . . . , Ps,z)}

  where :
  - z1, . . . , zs are global variables of *prog*;
  - P1, . . . , Ps are parameters;
  - z is a variable in {z1, . . . , zs};
  - **pre** and **f** are defined by a set of CHCs;
  - **f** is a functional relation: z=F(P1, . . . , Ps) for some function F defined for all P1, . . . , Ps that satisfy **pre**.

- All computable functions on integers can be specified by sets of CHCs.

Fibonacci Specification

- Fibonacci specification:

  {n=N, N>=0, u=1, v=0, t=0} *fibonacci* {fib(N,u)}

  where:

  fib(0,F) :- F=1.

  fib(1,F) :- F=1.

  fib(N3,F3) :- N1>=0, N2=N1+1, N3=N2+1, F3=F1+F2,
  fib(N1,F1), fib(N2,F2).

Translating Partial Correctness into CHCs

- A functional Horn specification can be translated into CHCs in two steps:

  Step1. Translate the operational semantics into CHCs;

  Step 2. Generate verification conditions as a set of CHCs.

Translating the Operational Semantics

- Define a relation **r_prog**(P1, . . . , Ps,Z) such that for all integers P1, . . . , Ps,

  if the initial values of the program variables satisfy the precondition **pre**(P1, . . . , Ps)

  then the final value of z computed by **prog** is Z.

- Fibonacci Example: Define a relation **r_fibonacci**(N,U) such that, for all integers N, if the program variables satisfy the precondition

  n=N, N>=0, u=1, v=0, t=0

  then the final value of u computed by program *fibonacci* is U.

… Translating the Operational Semantics

- **r_fibonacci** is defined by a set *OpSem* of clauses that encode the operational semantics:

```
r_fibonacci(N,U) :- initConf(Cf0,N), reach(Cf0,Cfh), finalConf(Cfh,U).
initConf(cf(LC,E),N) :- N>=0, U=1, V=0, T=0,
                        firstComm(LC),
                        env((n,N),E), env((u,U),E), env((v,V),E), env((t,T),E).
finalConf(cf(LC,E),U) :- haltComm(LC), env((u,U),E).
reach(Cf,Cf).
reach(Cf0,Cf2) :- tr(Cf0,Cf1), reach(Cf1,Cf2).
```

tr(Cf0,Cf1) is the transition relation that defines the interpreter of the imperative language [Peralta-Gallagher 98].

Generating Verification Conditions

- Verification conditions: Goals whose satisfiability guarantees that all clauses of the postcondition are satisfied by **r_prog**.

- For each clause **f**(P1, . . . ,Ps, Z) :- B defining the postcondition,

  (1) Replace **f** by **r_prog** in the head and in the body

  r_prog(P1, . . . ,Ps, Z) :- B′

  (2) Move the head atom to the body (exploiting functionality of **r_prog**):

  false :- Y≠Z, **r_prog**(P1, . . . ,Ps, Y), B′ where Y is a new variable

  (3) Case split:

  false :- Y>Z, **r_prog**(X1, . . . ,Xs, Z), B′

  false :- Y<Z, **r_prog**(X1, . . . ,Xs, Z), B′

- Theorem (Partial Correctness). ***If*** for all generated goals **false :- G**, *OpSem* ∪ {**false :- G**} is satisfiable, ***then*** the specification is valid.

Verification Conditions for Fibonacci

- Generating the verification conditions for *fibonacci*

fib(0,F) :- F=1.

(1) Replace **fib** by **r_fibonacci** in the head and in the body

r_fibonacci(0,F) :- F=1.

(2) Move the head atom to the body

false :- F≠1, r_fibonacci(0,F).

(3) Case split

G1: **false :- F>1, r_fibonacci(0,F).**

G2: **false :- F<1, r_fibonacci(0,F).**

Verification Conditions for Fibonacci

- Verification conditions for *fibonacci*

G1: **false :- F>1, r_fibonacci(0,F).**

G2: **false :- F<1, r_fibonacci(0,F).**

G3: **false :- F>1, r_fibonacci(1,F).**

G4: **false :- F<1, r_fibonacci(1,F).**

G5: **false :- N1>=0, N2=N1+1, N3=N2+1, F3>F1+F2,**
r_fibonacci(N1,F1), r_fibonacci(N2,F2), r_fibonacci(N3,F3).

G6: **false :- N1>=0, N2=N1+1, N3=N2+1, F3<F1+F2,**
r_fibonacci(N1,F1), r_fibonacci(N2,F2), r_fibonacci(N3,F3).

- Program *fibonacci* is partially correct if, for i=1,…,6, *OpSem* ∪ {Gi} is satisfiable.

Limitations of LA-solving

- Program *fibonacci* is partially correct and each *OpSem* ∪ {Gi} is satisfiable.

- However, there is no LA-solution for *OpSem* ∪ {G5} (and for *OpSem* ∪ {G6}).

  Proof (see details in the paper): there exists no LA constraint c(N,F) which is an LA-solution of the clauses for **r_fibonacci** and:

  LA ⊨ ∀(N1>=0, N2=N1+1, N3=N2+1, F3>F1+F2,
  c(N1,F1), c(N2,F2), c(N3,F3) → false)

- LA-solvers cannot prove the partial correctness of *fibonacci.*

Improving LA-solving by Transforming Verification Conditions

- Possible solution: More powerful constraint theories, but decidability of entailment is lost for non-linear polynomials [Matijasevic 70].

- Our solution: Transform the verification conditions into equisatisfiable CHCs, which are (hopefully) LA-solvable.

- Transformation = Removal of Interpreter; Linearization

Removal of the Interpreter

- Specialize *OpSem* w.r.t. *fibonacci* to avoid the interpretation overhead (hence improving the efficiency of LA-solving).

- $OpSem_{RI}$:

```
r_fibonacci(N,F) :-                                   %Initialization
        N>=0, U=1, V=0, T=0,
        r(N,U,V,T,  N1,F,V1,T1).
r(N,U,V,T, N2,U2,V2,T2) :-                            %Loop
        N>=1, N1=N-1, U1=U+V, V1=U, T1=U,
        r(N1,U1,V1,T1, N2,U2,V2,T2).
r(N,U,V,T,  N,U,V,T) :- N=<0.                         %Loop exit
```

- For each Gi, *OpSem* ∪ {Gi} is satisfiable ***iff*** $OpSem_{RI}$ ∪ {Gi} is satisfiable.

- No references to the interpreter (i.e., the transition relation) **tr** in $OpSem_{RI}$

Linearization

G5: **false :- N1>=0, N2=N1+1, N3=N2+1, F3>F1+F2,**
r_fibonacci(N1,F1), r_fibonacci(N2,F2), r_fibonacci(N3,F3).

- No LA-solution of single **r_fibonacci** atoms is able to prove that the body of G5 is false.
- An "LA-solution" for the conjunction of the three **r_fibonacci** atoms exists. The relation among the three atoms is linear:
  N1>=0, N2=N1+1, N3=N2+1, F3=F1+F2
- Transform conjunctions of atoms into single atoms (and then apply LA-solving), i.e., transform $OpSem_{RI} \cup \{G5\}$ into a set of linear clauses:

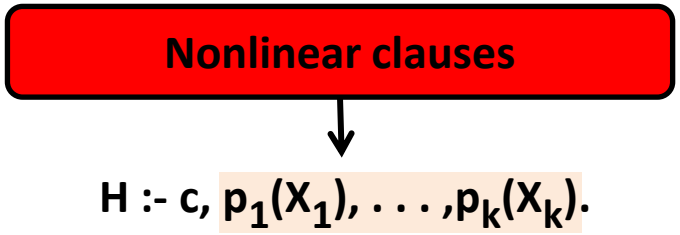
  $\mathbf{A_0}$ **:-** $\mathbf{c}$, $\mathbf{A_1}$**.**

  where $\mathbf{A_0}$ is either an atom or **false**, and $\mathbf{A_1}$ is either an atom or **true**.
- The transformation makes use of the fold/unfold rules [Tamaki-Sato 84, Etalle-Gabbrielli 96].

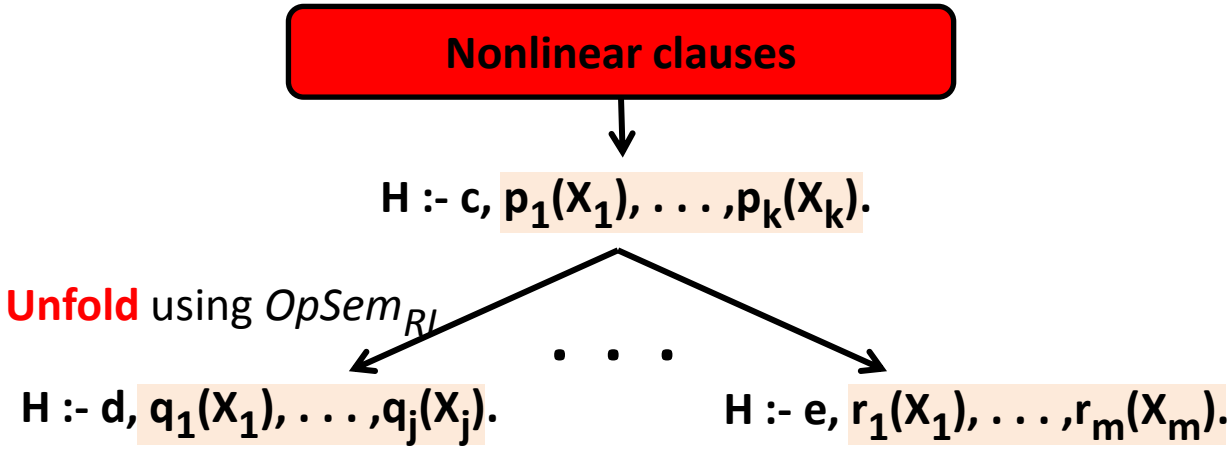
The Linearization Transformation

Nonlinear clauses

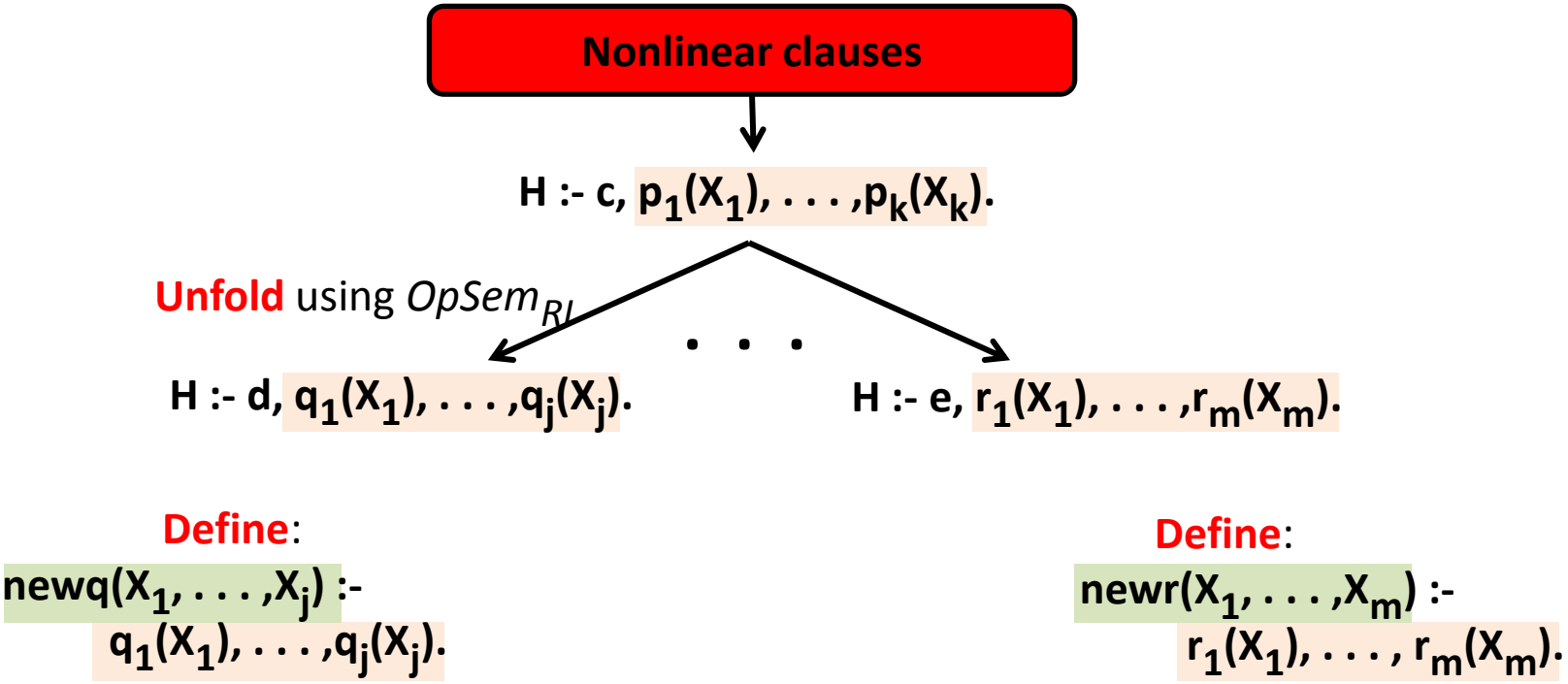
The Linearization Transformation



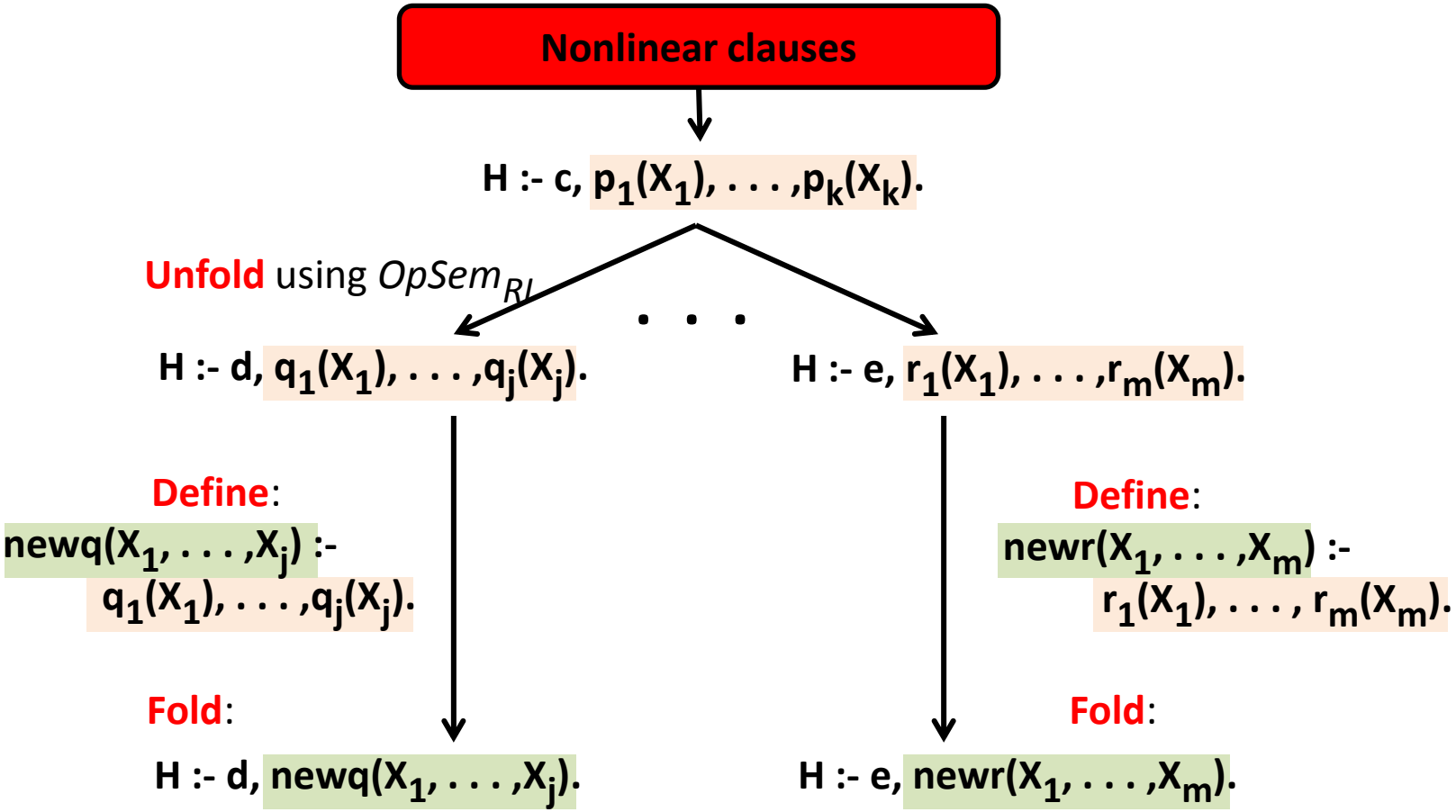
The Linearization Transformation



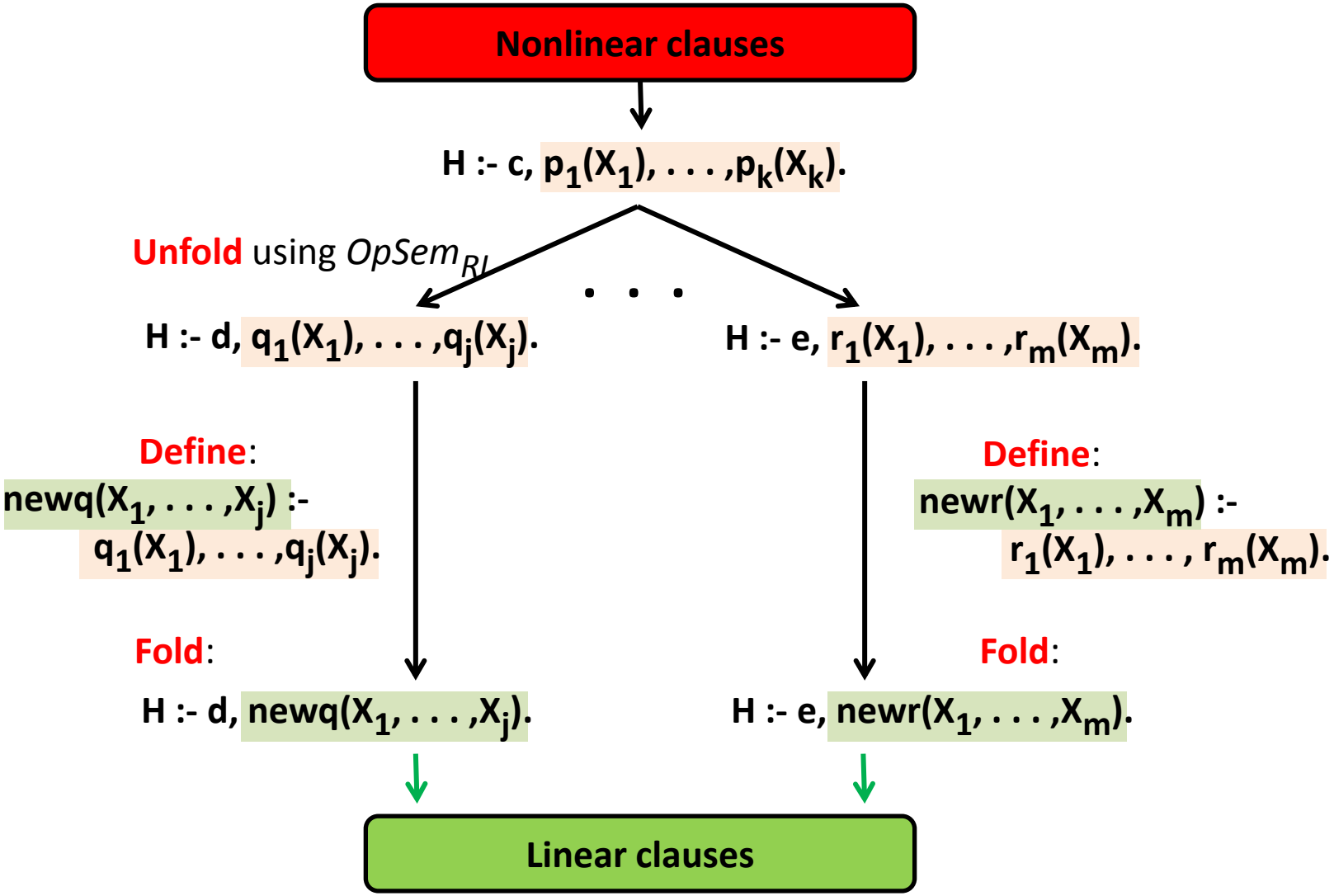
The Linearization Transformation



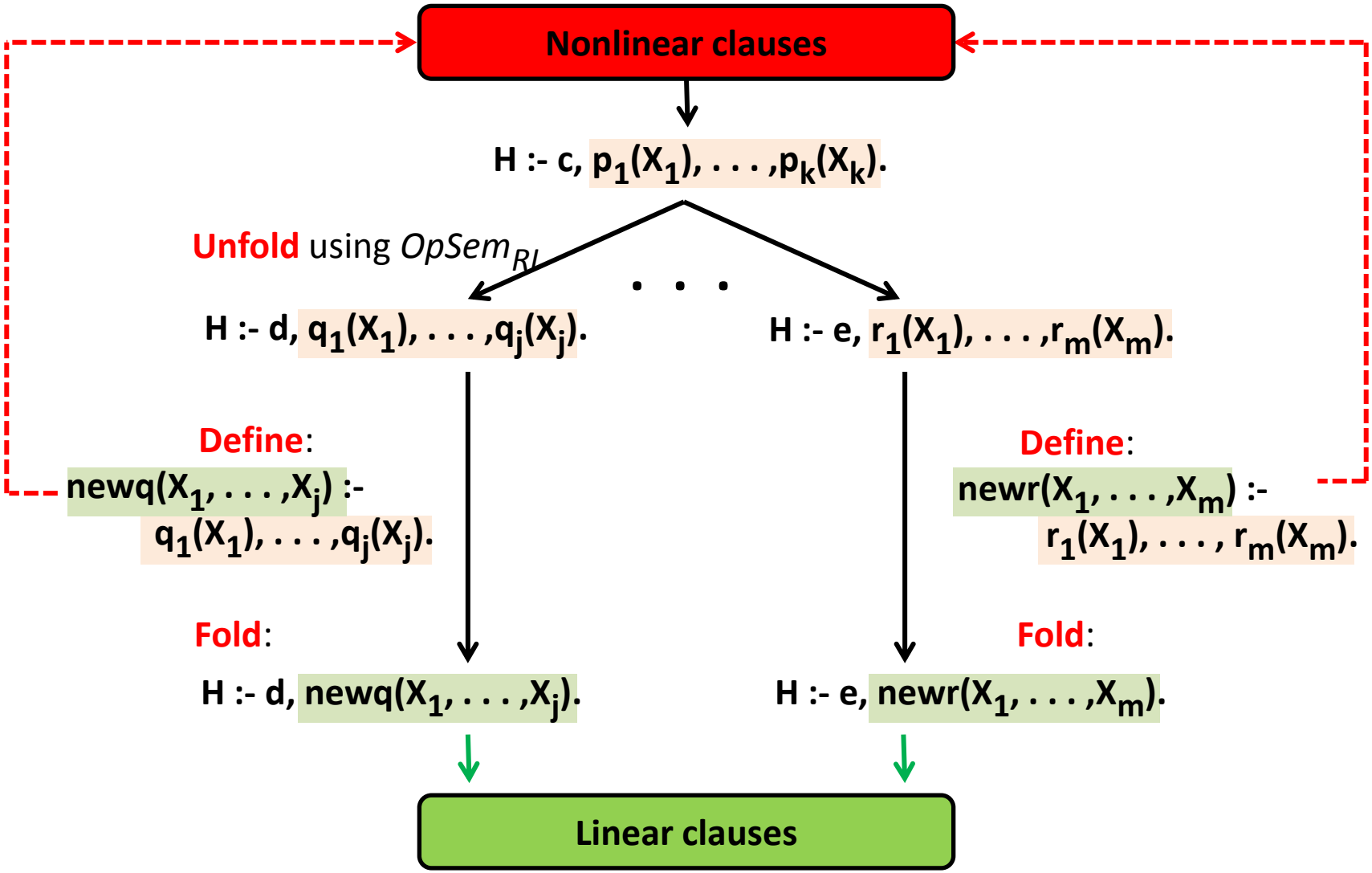
The Linearization Transformation



The Linearization Transformation



The Linearization Transformation



Linearized Fibonacci

false :- N1>= 0, N2=N1+1, N3=N2+1, F3>F1+F2, U=1, V=0, **new1**(N3,U,V,F3,N2,F2,N1,F1).
new1(N1,U,V,U,N2,U,N3,U) :- N1=<0, N2=<0, N3=<0.
new1(N1,U,V,U,N2,U,N3,F3) :- N1=<0, N2=<0, N4=N3-1, W=U+V, N3>=1, **new2**(N4,W,U,F3).
new1(N1,U,V,U,N2,F2,N3,U) :- N1=<0, N4=N2-1, W=U+V, N2>=1, N3=<0, **new2**(N4,W,U,F2).
new1(N1,U,V,U,N2,F2,N3,F3) :- N1=<0, N4=N2-1, N2>=1, N5=N3-1, N3>=1, **new3**(N4,W,U,F2,N5,F3).
new1(N1,U,V,F1,N2,U,N3,U) :- N4=N1-1, W=U+V, N1>=1, N2=<0, N3=<0, **new2**(N4,W,U,F1).
new1(N1,U,V,F1,N2,U,N3,F3) :- N4=N1-1, N1>=1, N2=<0, N5=N3-1, W=U+V, N3>=1, **new3**(N4,W,U,F1,N5,F3).
new1(N1,U,V,F1,N2,F2,N3,U) :- N4=N1-1, N1>=1, N5=N2-1, W=U+V, N2>=1, N3=<0, **new3**(N4,W,U,F1,N5,F2).
new1(N1,U,V,F1,N2,F2,N3,F3) :- N4=N1-1, N1>=1, N5=N2-1, N2>=1, N6=N3-1, W=U+V, N3>=1, **new1**(N4,W,U,F1,N5,F2,N6,F3).
new2(N,U,V,U) :- N=<0.
new2(N,U,V,F) :- N2=N-1, W=U+V, N>=1, **new2**(N2,W,U,F).
new3(N1,U,V,U,N2,U) :- N1=<0, N2=<0.
new3(N1,U,V,U,N2,F2) :- N1=<0, N3=N2-1, W=U+V, N2>=1, **new2**(N3,W,U,F2).
new3(N1,U,V,F1,N2,F2) :- N3=N1-1, N1>=1, N4=N2-1, W=U+V, N2>=1, **new3**(N3,W,U,F1,N4,F2).
new3(N1,U,V,F1,N2,U) :- N3=N1-1, W=U+V, N1>=1, N2=<0, **new2**(N3,W,U,F1).

where **new1**, **new2**, **new3** have been introduced by the following **definitions**:

new1(N1,U,V,F1,N2,F2,N3,F3) :- r(N1,U,V,V,X1,F1,Y1,Z1), r(N2,U,V,V,X2,F2,Y2,Z2), r(N3,U,V,V,X3,F3,Y3,Z3).
new2(N,U,V,F) :- r(N,U,V,V,X,F,Y,Z).
new3(N2,U,V,F2,N1,F1) :- r(N1,U,V,V,X1,F1,Y1,Z1), r(N2,U,V,V,X2,F2,Y2,Z2).

The linearized clauses for *fibonacci* are LA-solvable.

Properties of the Transformation

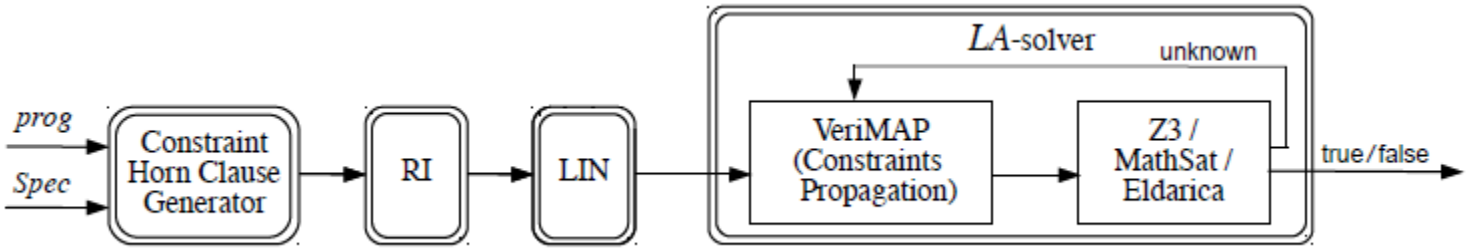
- $OpSem_{RI}$ is a set of linear clauses (no recursive functions in the imperative language).

- For every goal **false :- G**, the linearization transformation terminates for the input
  $OpSem_{RI} \cup \{$**false :- G**$\}$

- Let *TransfCls* be the output of linearization.
  $OpSem_{RI} \cup \{$**false :- G**$\}$ is satisfiable ***iff*** *TransfCls* is satisfiable.

- ***If*** $OpSem_{RI} \cup \{$**false :- G**$\}$ is LA-solvable,
  then *TransfCls* is LA-solvable.
  Not vice versa: LA-solvability can be increased.

Implementation



Available at **http://map.uniroma2.it/linearization**

Experiments

Program	RI	*LA*-solving-1		LIN	*LA*-solving-2: VeriMAP &		
		Z3	Eldarica		Z3	MathSAT	Eldarica
1. *binary_division*	0.02	4.16	*TO*	0.04	17.36	17.87	20.98
2. *fast_multiplication_2*	0.02	*TO*	3.71	0.01	1.07	1.97	7.59
3. *fast_multiplication_3*	0.03	*TO*	4.56	0.02	2.59	2.54	9.31
4. *fibonacci*	0.01	*TO*	*TO*	0.01	2.00	47.74	6.97
5. *Dijkstra_fusc*	0.01	1.02	3.80	0.05	2.14	2.80	10.26
6. *greatest_common_divisor*	0.01	*TO*	*TO*	0.01	0.89	1.78	0.04
7. *integer_division*	0.01	*TO*	*TO*	0.01	0.88	1.90	2.86
8. 91-*function*	0.01	1.27	*TO*	0.06	117.97	14.24	*TO*
9. *integer_multiplication*	0.02	*TO*	*TO*	0.01	0.52	14.76	0.54
10. *remainder*	0.01	*TO*	*TO*	0.01	0.87	1.70	3.16
11. *sum_first_integers*	0.01	*TO*	*TO*	0.01	1.79	2.30	6.81
12. *lucas*	0.01	*TO*	*TO*	0.01	2.04	8.39	9.46
13. *padovan*	0.01	*TO*	*TO*	0.01	2.24	*TO*	11.62
14. *perrin*	0.01	*TO*	*TO*	0.02	2.23	*TO*	11.89
15. *hanoi*	0.01	*TO*	*TO*	0.01	1.81	2.07	6.59
16. *digits*10	0.01	*TO*	*TO*	0.01	4.52	3.10	6.54
17. *digits*10-*itmd*	0.06	*TO*	*TO*	0.04	*TO*	10.26	12.38
18. *digits*10-*opt*	0.08	*TO*	*TO*	0.10	*TO*	*TO*	15.80
19. *digits*10-*opt*100	0.01	*TO*	*TO*	0.02	*TO*	58.99	8.98

Table 1. *Columns* RI *and* LIN *show the times (in seconds) taken for removal of the interpreter and linearization. The two columns under LA-*solving-1 *show the times taken by* Z3 *and* Eldarica *for solving the problems after* RI *alone. The three columns under LA-*solving-2 *show the times taken by* VeriMAP *together with* Z3, MathSAT, and Eldarica, *after* RI *and* LIN. *The timeout TO occurs after* 120 *seconds.*

Conclusions

- LA-solving < Linearization; LA-solving
- Key idea: Use of conjunctive definitions and folding.
- Future work:
  - Design a smarter linearization for deriving “better” CHCs (smaller, easier to solve);
  - Consider richer imperative languages (recursion, arrays, structures, pointers, etc.) ;
  - Investigate the relations among satisfiability , **C**-solvability, and transformations for other classes **C** of constraints.