Issues, Solutions and Tools in

Formal Proofs for the TMD

Érik Martin-Dorel, Micaela Mayero, Ioana Paşca,
Laurence Rideau, Laurent Théry

Reunion TaMaDi - Février 2011

How do we make formal proofs?

Example:

- prove correct an algorithm for computing roots of a function

How do we make formal proofs?

Example:

- prove correct an algorithm for computing roots of a function

Solutions:

- formalize everything in Coq
  - implement the algorithm in Coq
  - prove its correctness

How do we make formal proofs?

Example:

- prove correct an algorithm for computing roots of a function

Solutions:

- use annotations for the actual code
  - implement the algorithm in C (for example)
  - write annotation for the behavior of the algorithm
  - generate proof obligations and prove them in Coq

How do we make formal proofs?

Example:

- prove correct an algorithm for computing roots of a function

Solutions:

- use certificates
  - run the algorithm (possibly outside Coq)
  - check in Coq that the result is the root of the function

How do we make formal proofs?

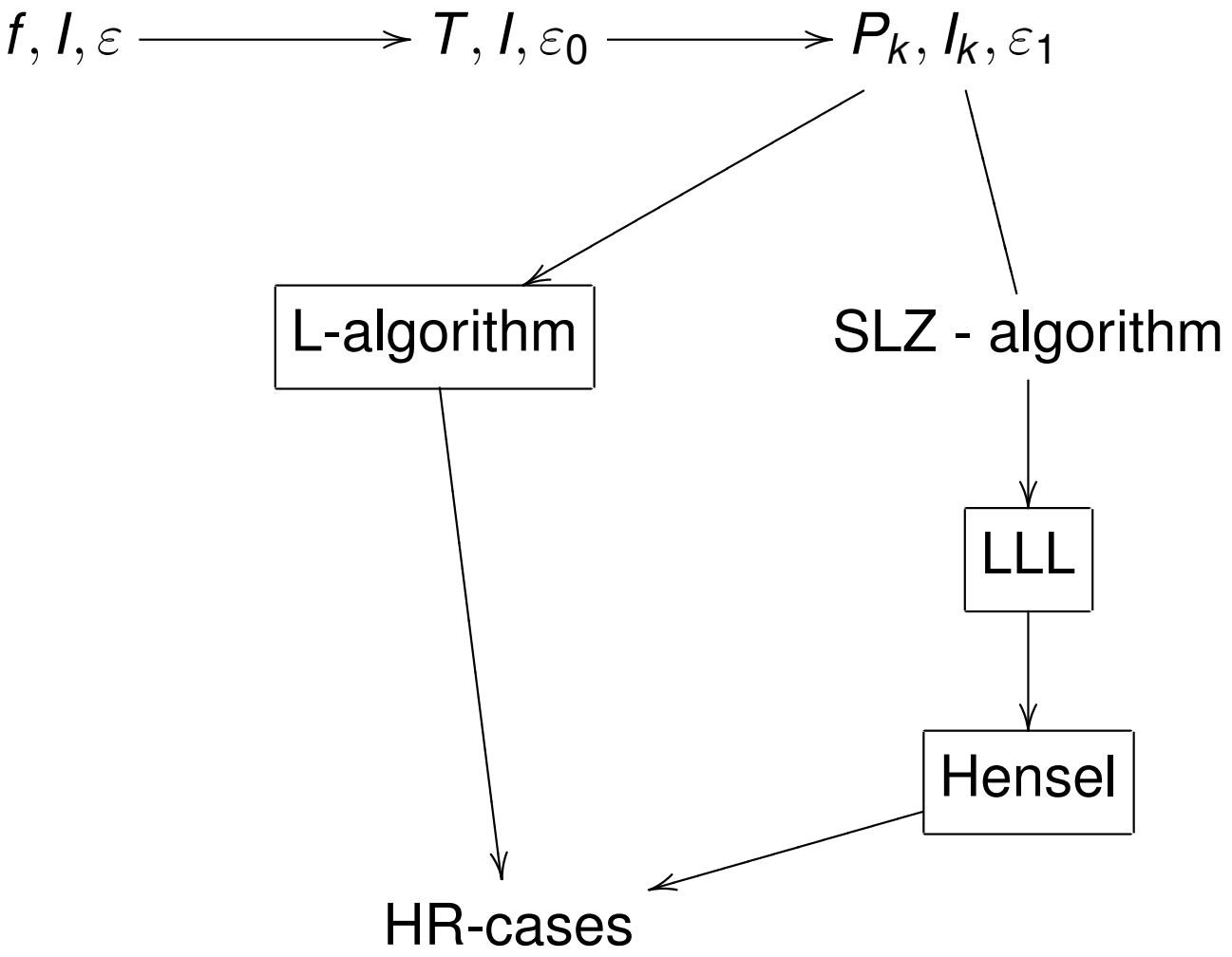
Example:

- prove correct an algorithm for computing roots of a function

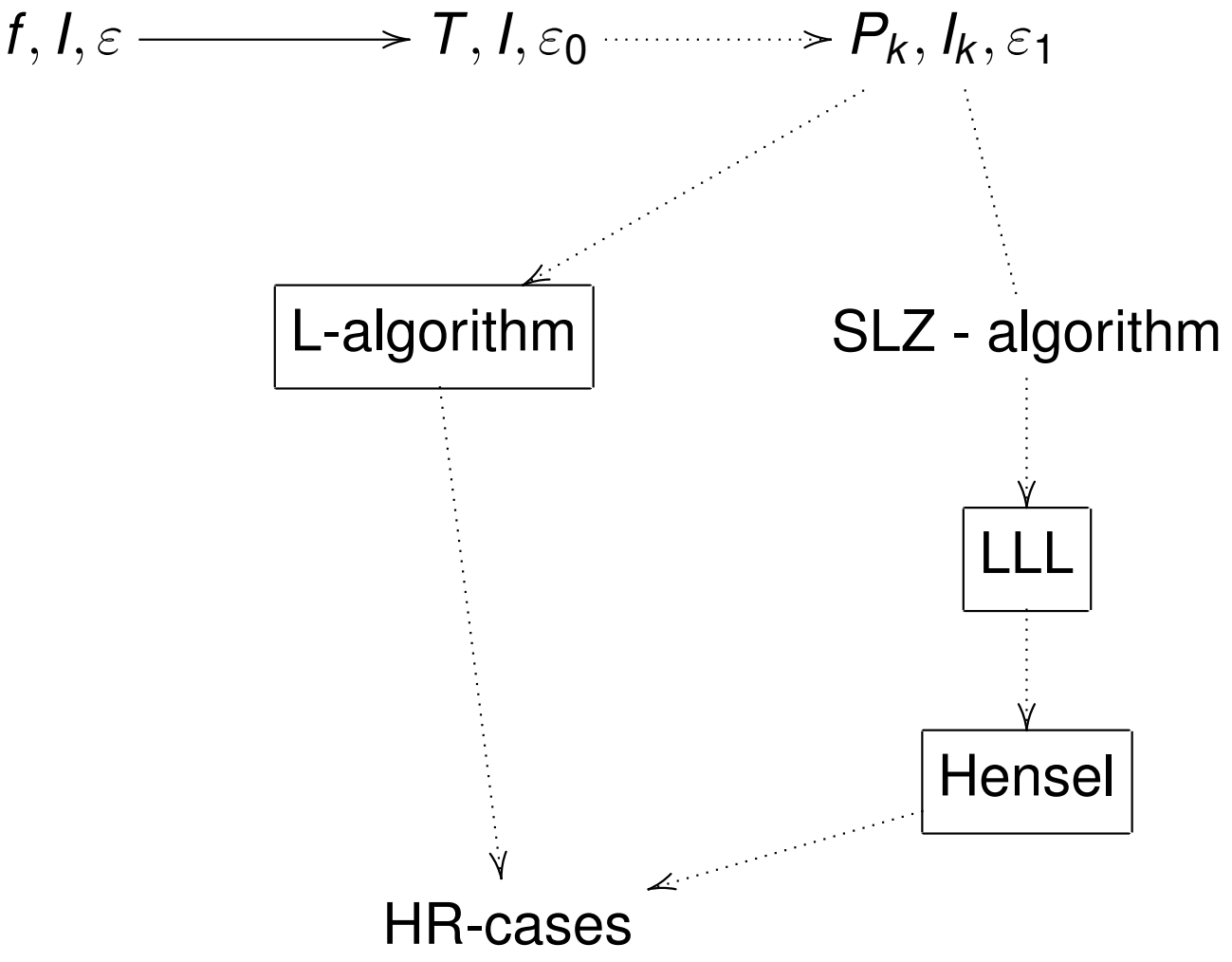
Solutions:

- formalize everything in Coq
  - implement the algorithm in Coq
  - prove its correctness
- use annotations for the actual code
  - implement the algorithm in C (for example)
  - write annotation for the behavior of the algorithm
  - generate proof obligations and prove them in Coq
- use certificates
  - run the algorithm (possibly outside Coq)
  - check in Coq that the result is the root of the function

A formal proof view of TMD



A formal proof view of TMD



I. Polynomial approximations in Coq

$$f, I, \varepsilon \xrightarrow[\text{Models}]{\text{Taylor}} T^{\{n\}}, I, \varepsilon_0$$

I. Polynomial approximations in Coq

$$f, I, \varepsilon \xrightarrow[\text{Models}]{\text{Taylor}} T^{\{n\}}, I, \varepsilon_0$$

Task:

- formally certify that the computed polynomial $T^{\{n\}}$ satisfies $\|f - T^{\{n\}}\| \leq \varepsilon_0$ on I

I. Polynomial approximations in Coq

$$f, I, \varepsilon \xrightarrow[\text{Models}]{\text{Taylor}} T^{\{n\}}, I, \varepsilon_0$$

Task:

- formally certify that the computed polynomial $T^{\{n\}}$ satisfies $\|f - T^{\{n\}}\| \leq \varepsilon_0$ on I

Issues:

- certificate based approach does not work
- computation on intervals in Coq is not accurate enough

I. Polynomial approximations in Coq

$$f, I, \varepsilon \xrightarrow[\text{Models}]{\text{Taylor}} T^{\{n\}}, I, \varepsilon_0$$

Task:

- formally certify that the computed polynomial $T^{\{n\}}$ satisfies $\|f - T^{\{n\}}\| \leq \varepsilon_0$ on I

Issues:

- certificate based approach does not work
- computation on intervals in Coq is not accurate enough

What solutions?

- formalize Taylor Model algorithm in Coq
- use theoretical properties of polynomial T

Taylor Models (TM)

a TM for f on the interval I is a pair $(T; \Delta)$

- T, polynomial
- Δ, interval
- $\forall x \in I, f(x) - T(x) \in \Delta$

$$T(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$
$$\Delta = \frac{f^{(n+1)}(I)}{n!}(I - x_0)^{n+1}$$

Advantages:

- easy to compute coefficients of T
- easy to compute Δ as the Lagrange reminder
- tight bound for the reminder
- well-studied theoretical properties

Coq formalization of TM

Goal:

- be able to compute T and Δ in Coq
- be able to prove the result is correct, that is

$$\forall x \in I, f(x) - T(x) \in \Delta$$

Libraires for computation in Coq

exact real numbers

- \+ all basic functions
- \+ simpler proofs
- – no evaluation on intervals
- – efficiency ?

FP numbers and intervals

- \+ closer to the real world implementation
- \+ evaluation on intervals
- – more complicated proofs
- – not completely formally proved

Libraires for computation in Coq

exact real numbers

- \+ all basic functions
- \+ simpler proofs
- – no evaluation on intervals
- – efficiency ?

FP numbers and intervals

- \+ closer to the real world implementation
- \+ evaluation on intervals
- – more complicated proofs
- – not completely formally proved

For exp, on intervals of length 2^{-10}, poly of degree 10:

- exact real numbers 2-3 sec/error computation
- interval FP numbers 1-2 sec/error computation

What choice?

For the library:

- use libraries on FP numbers and intervals

For the Taylor polynomial:

- use interval coefficients ?
- use FP coefficients ?

What about the proofs?

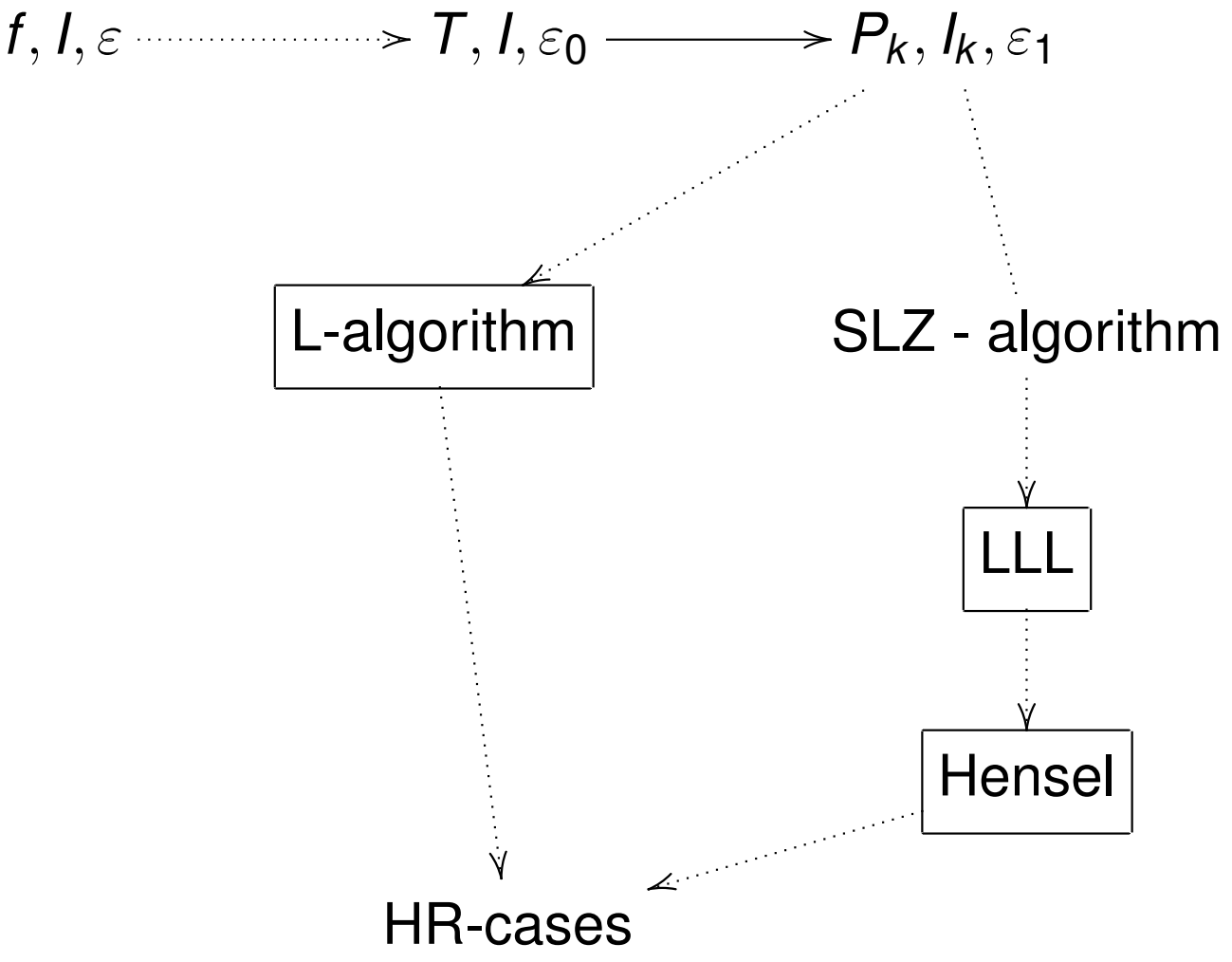
work done for Taylor Models by R. Zumkeller

- remainder as interval with exact real bounds
- no models for basic functions (?)

work for TaMaDi

- prove background results
  - Taylor's theorem
  - correctness of functions in the FP interval library
- formalize and prove correct Taylor models
  - with FP or FP interval coefficients and FP interval reminder

A formal proof view of TMD



II. Hierarchical approximations

$$T^{\{n\}}, I, \varepsilon_0 \xrightarrow[\text{approximation}]{\text{Hierarchical}} P_k^{\{1,2,3\}}, I_k, \varepsilon_1$$

II. Hierarchical approximations

$$T^{\{n\}}, I, \varepsilon_0 \xrightarrow[\text{approximation}]{\text{Hierarchical}} P_k^{\{1,2,3\}}, I_k, \varepsilon_1$$

Task:

- formally verify that $\| T^{\{n\}} - P_k^{\{1,2,3\}} \| \leq \varepsilon_1$ on each I_k

II. Hierarchical approximations

$$T^{\{n\}}, I, \varepsilon_0 \xrightarrow[\text{approximation}]{\text{Hierarchical}} P_k^{\{1,2,3\}}, I_k, \varepsilon_1$$

Task:

- formally verify that $\| T^{\{n\}} - P_k^{\{1,2,3\}} \| \leq \varepsilon_1$ on each I_k

Issues:

- a LOT of intervals I_k

II. Hierarchical approximations

$$T^{\{n\}}, I, \varepsilon_0 \xrightarrow[\text{approximation}]{\text{Hierarchical}} P_k^{\{1,2,3\}}, I_k, \varepsilon_1$$

Task:

- formally verify that $\| T^{\{n\}} - P_k^{\{1,2,3\}} \| \leq \varepsilon_1$ on each I_k

Issues:

- a LOT of intervals I_k

What solutions?

- sum-of-squares technique (see presentation tomorrow)
- verify the algorithm that computes P_k from T (see presentation today)

Solution 1: sum-of-squares

- certificate based approach

Idea:

$$T^{\{n\}}(X) - P_k^{\{1,2,3\}(X)} = \sum_{i=0}^{n} c_i X^i$$

$$\sum_{i=0}^{n} c_i X^i \leq \varepsilon_1 \Leftrightarrow \varepsilon_1 - \sum_{i=0}^{n} c_i X^i \geq 0 \Leftrightarrow \sum_{i=0}^{n} d_i X^i \geq 0$$

- write this quantity as a sum of squares
- decomposition done by an external tool
- decomposition checked in Coq

Solution 2: verify the algorithm

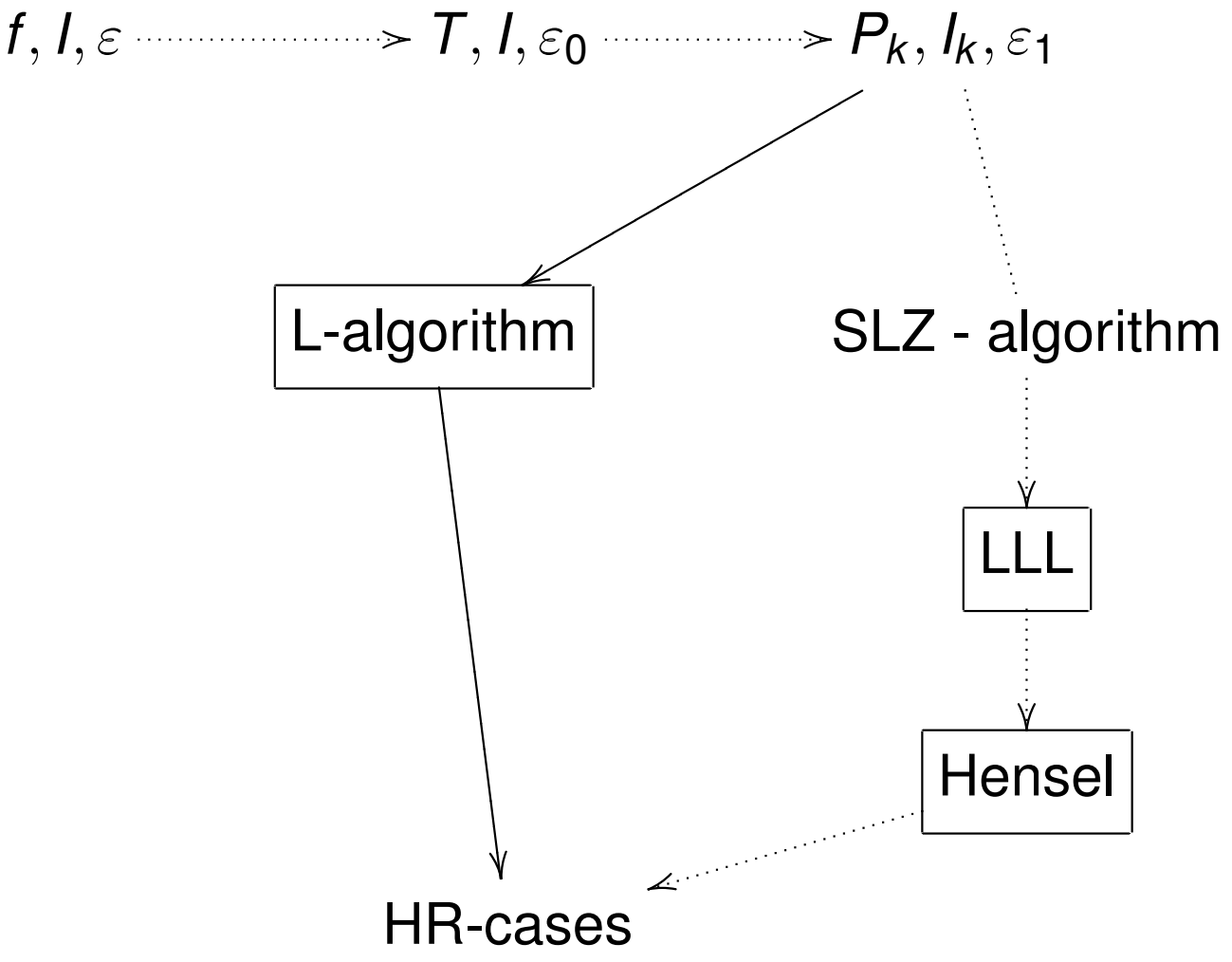
$$T^{\{n\}}(X) = \sum_{i=0}^{n} a_i X^i$$

- we obtain the coefficients of $P_k^{\{1,2,3\}}$ as $h(a_0, \ldots, a_n)$
- we obtain the error $\| T^{\{n\}} - P_k^{\{1,2,3\}} \| \leq \varepsilon_1(a_0, \ldots, a_n)$

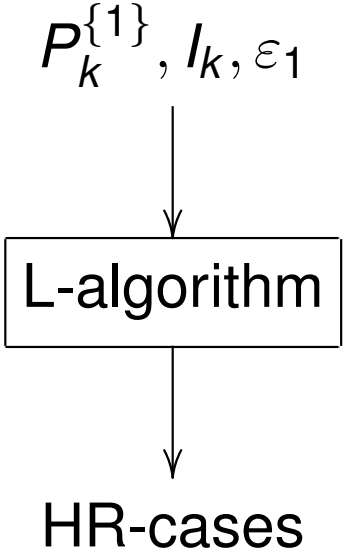
Idea:

- verify these steps in Coq
- evaluate the error directly

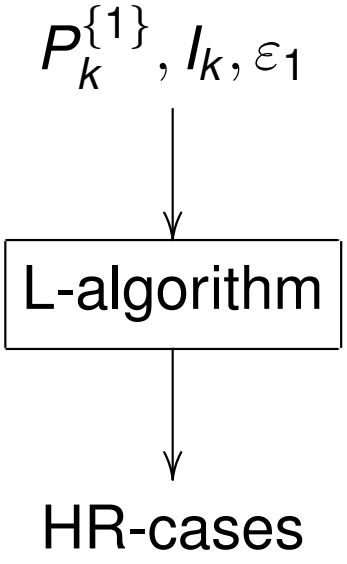
A formal proof view of TMD



III. The L-algorithm



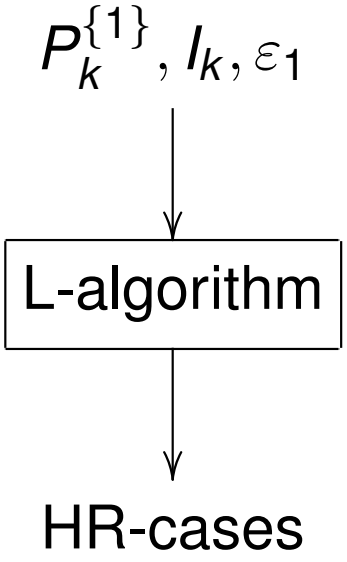
III. The L-algorithm



Task:

- formally prove the L-algorithm outputs
  - all HR-cases or
  - absence of HR-cases

III. The L-algorithm



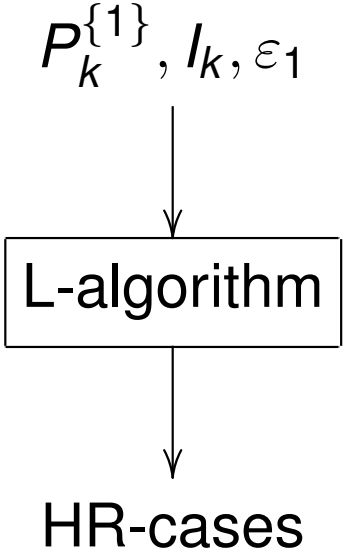
Task:

- formally prove the L-algorithm outputs
  - all HR-cases or
  - absence of HR-cases

Issues:

- L-algorithm is highly optimized in practice
- $P_k(x) = a \cdot x + b$, with a - irrational

III. The L-algorithm



Task:

- formally prove the L-algorithm outputs
  - all HR-cases or
  - absence of HR-cases

Issues:

- L-algorithm is highly optimized in practice
- $P_k(x) = a \cdot x + b$, with a - irrational

What solutions?

- prove the annotated C code
- re-think the proof

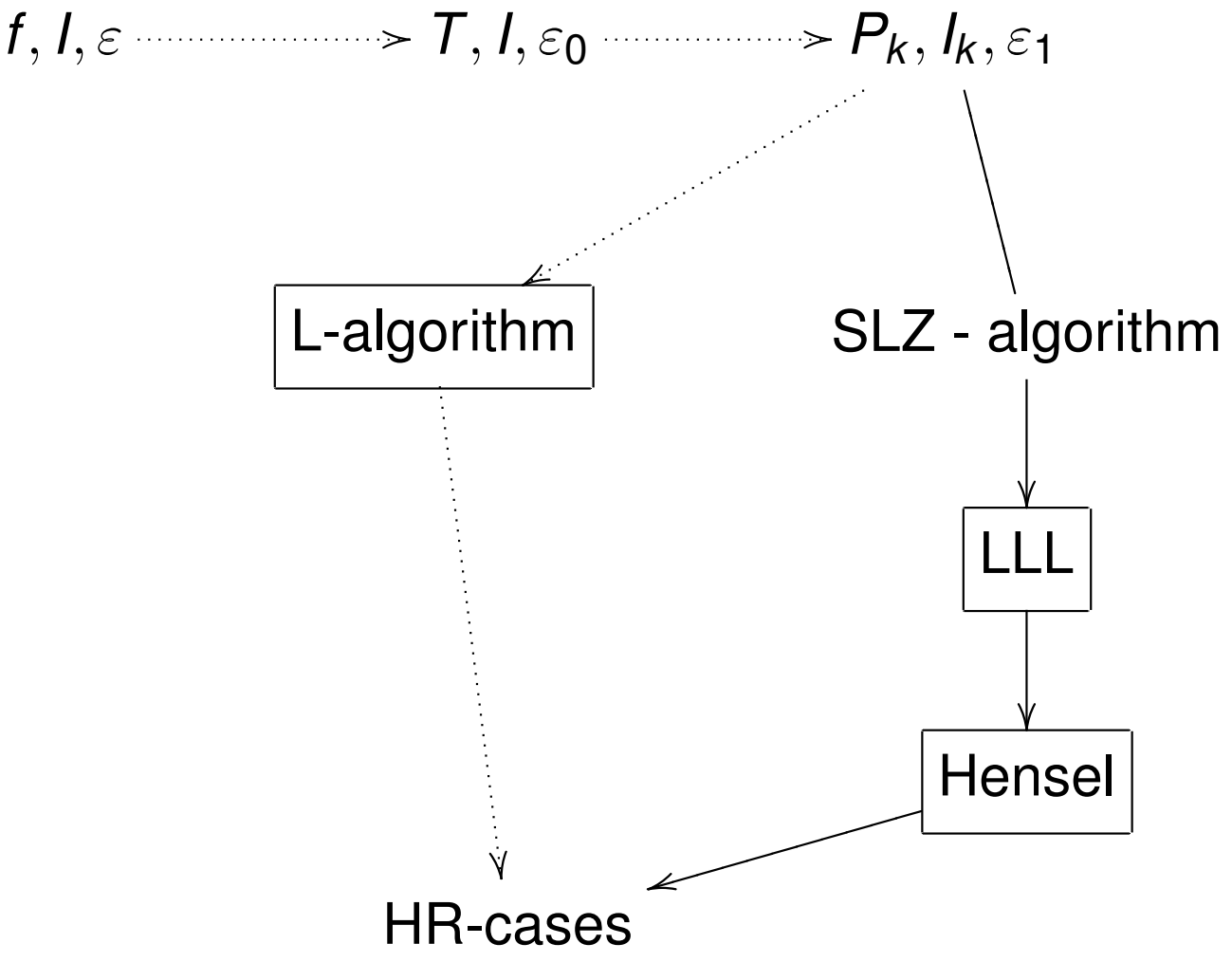
How do we prove C code?

```
/*@ ensures \result >= x && \result >= y;
    ensures \result == x || \result == y;
*/
int max (int x, int y) { return (x > y) ? x : y; }
```

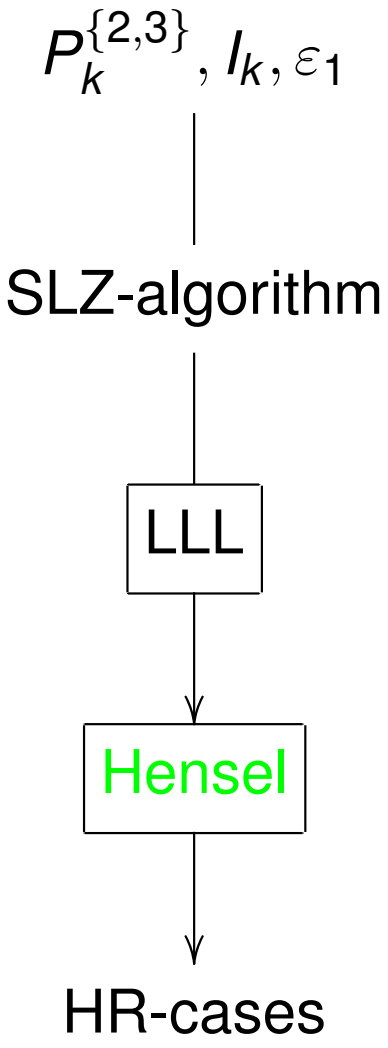
Frama-C framework

- take annotated C code
- generate proof obligations
- attempt to solve them by automatic and interactive provers

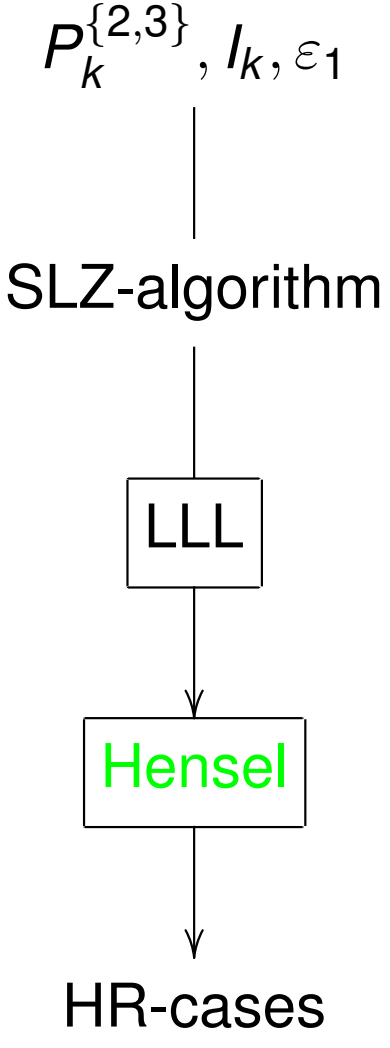
A formal proof view of TMD



IV. The SLZ-algorithm



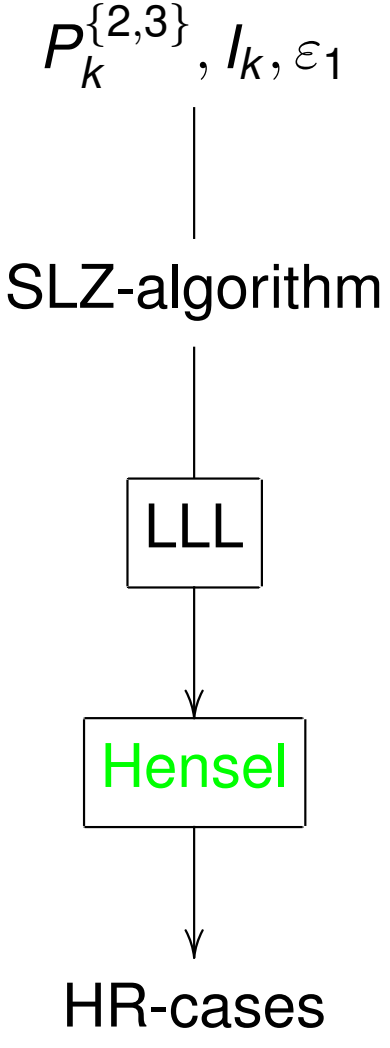
IV. The SLZ-algorithm



Task:

- formally prove the algorithm outputs
  - all HR-cases or
  - absence of HR-cases

IV. The SLZ-algorithm



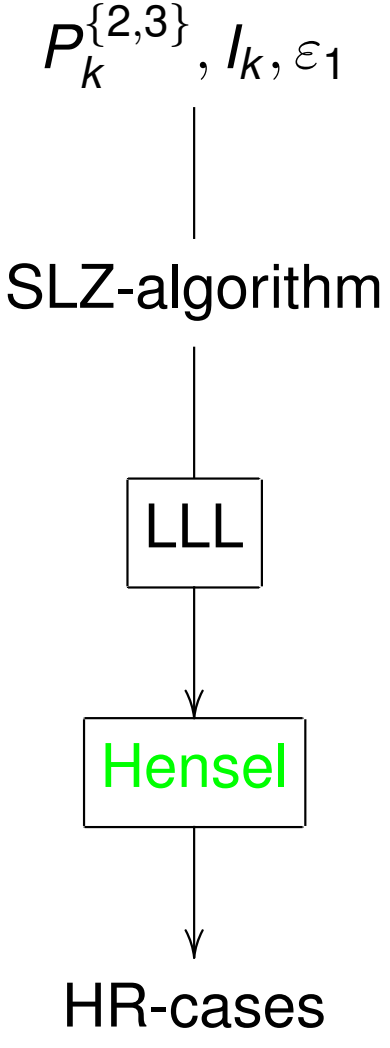
Task:

- formally prove the algorithm outputs
  - all HR-cases or
  - absence of HR-cases

Issues:

- LLL is complicated

IV. The SLZ-algorithm



Task:

- formally prove the algorithm outputs
  - all HR-cases or
  - absence of HR-cases

Issues:

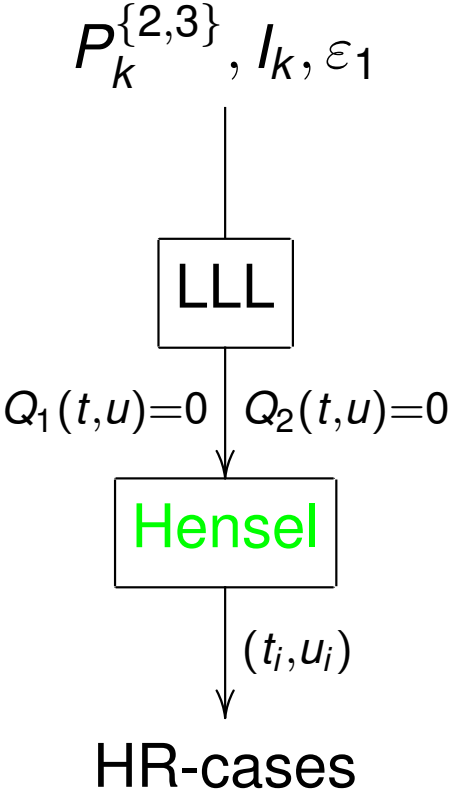
- LLL is complicated

What solutions?

- use a certificate based approach
- see paper by Érik (and talk at the first TaMaDi reunion)

A quick reminder on Hensel's lifting

compute all integer roots of a system of 2 bivariate polynomials with integer coefficients



What next?

I. What certification polynomial approximations?

- certify Sollya polynomial
- use Coq polynomial (in this case we can)
- are they the same (or close)?

What next?

I. What certification polynomial approximations?
   - certify Sollya polynomial
   - use Coq polynomial (in this case we can)
   - are they the same (or close)?

II. How will hierarchical approximation certification work?
   - we cannot use Coq results directly (far too slow)
   - can we prove optimized code?

What next?

I. What certification polynomial approximations?
- certify Sollya polynomial
- use Coq polynomial (in this case we can)
- are they the same (or close)?

II. How will hierarchical approximation certification work?
- we cannot use Coq results directly (far too slow)
- can we prove optimized code?

III. The L-algorithm
- also highly optimized code
- can we simplify the proof for the (paper) algorithm?

What next?

I. What certification polynomial approximations?
- certify Sollya polynomial
- use Coq polynomial (in this case we can)
- are they the same (or close)?

II. How will hierarchical approximation certification work?
- we cannot use Coq results directly (far too slow)
- can we prove optimized code?

III. The L-algorithm
- also highly optimized code
- can we simplify the proof for the (paper) algorithm?

IV. SLZ-algorithm
- complete certificate