Specifying the roundoff errors of SONNX components

Franck Vedrine – CEA
with the help of Eric Jenn – IRT Saint-Exupery

Accumulation of roundoff errors for SONNX tensor algorithms

- Numerous possible numerical format for computations
  - floating-point: double, float, bfloat16, …
  - fixed-point: different sizes for the fractional part
- Numerous possible algorithms
  - Ex: optimizations for SoftMax
  - With Error Free Transformation (Rump & all) techniques to gain precision
    - Ex: an EFT for Addition: FastTwoSum, a + b = s + t

```
void FastTwoSum(double a, double b, double* s, double* t)
{ // No unsafe optimizations!
  *s = a + b ;
  *t = b - (* s - a );
}
```

- How to define **precise** and **concise** accuracy formula that are **verifiable for each implementation**?

Available technologies

- Arithmetic for the accumulation of roundoff error
  - Concrete comparison/difference between the implementation format and a better data format
  - Stochastic arithmetic – Cadna (source instrumentation - LIP6), Verrou (binary instrumentation with valgrind - EDF), Verificarlo (intermediary instrumentation with llvm - UVSQ)
  - Affine arithmetic – Fluctuat (abstract interpreter - CEA), FLDLib (source instrumentation, “abstract compiler” - CEA) – soundness
  - Formal arithmetic and simplification – soundness

- in different analysis technologies
  - Instrumentation of floating-point data
  - Abstract interpretation – all numerical data
  - Instrumentation of all numerical data

Inference rules of atomic operations

- Every concrete value f is defined as $x + e$ where
  - x is a real number that would be computed if the numerical format was ideal
  - e is a real number representing the numerical error that is the difference between f the floating-point or fixed-point value and x
  - $+, -, \times$, / are operations in real numbers, $+_{\text{impl}}, -_{\text{impl}}, *_{\text{impl}}, /_{\text{impl}}$ are operations in floating/fixed-point

- The **induced error** is a function taking an input error e and returning how it is amplified
- The **introduced error** is an new error that the operation creates to approximate the ideal result in a storing objective

With these definitions, the inference rules are

- $-_{\text{impl}}(x + e) = -x + \boldsymbol{-e}$ is an exact operation
- $(x_0 + e_0) +_{\text{impl}} (x_1 + e_1) = (x_0 + x_1) + (\boldsymbol{e_0 + e_1}) +$ introduced error of $\left((x_0 + x_1) + (e_0 + e_1)\right)$
- $(x_0 + e_0) -_{\text{impl}} (x_1 + e_1) = (x_0 - x_1) + (\boldsymbol{e_0 - e_1}) +$ introduced error of$\left((x_0 - x_1) + (e_0 - e_1)\right)$
- $(x_0 + e_0) *_{\text{impl}} (x_1 + e_1) = (x_0 \times x_1) + \left(\boldsymbol{e_0 \times x_1 + e_1 \times (x_0 + e_0)}\right) +$ introduced error of$\left((x_0 \times x_1) + \left(e_0 \times x_1 + e_1 \times (x_0 + e_0)\right)\right)$
- $(x_0 + e_0) /_{\text{impl}} (x_1 + e_1) = \frac{x_0}{x_1} + \boldsymbol{\frac{x_0 \times e_1 - e_0 \times x_1}{x_1 \times (x_1 + e_1)}} +$ introduced error of $\left(\frac{x_0}{x_1} + \frac{x_0 \times e_1 - e_0 \times x_1}{x_1 \times (x_1 + e_1)}\right)$

Focus on the introduced error

- In floating-point, the introduced error of (x)
  - $\leq |x| \times u$, where u is the semi-ulp of 1, that is 2^{-53} for the double type
  - could be improved with the Sterbenz Lemma |a| $-_{\text{impl}}$ |b| is exact if $\frac{1}{2}|a| \leq |b| \leq 2|a|$
    - this lemma needs the operation and a range for the operands
  - is deterministic in every mode and can be exactly computed if x is known (see EFT)

- In fixed-point, the introduced error of (x)
  - It often needs the operations and the operands
    - x $+_{\text{impl}}$ y does not introduce any error
  - $\leq u$, where u is given by the fractional part of the fixed-point format

Objectives for SONNX components

1. Write an accuracy specification for tensor operations that are verifiable
   - Tensor operations combine an imprecise (potentially unbound) number of atomic operations (dimensions)
   - The specification should allow multiple implementations

2. Check if an implementation verifies a specification with analyses ensuring some guarantees

3. Show that an implementation is better than another one from the accuracy point of view

Subjective approach/methodology

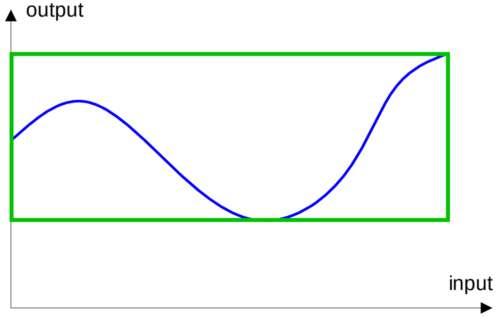
1. **Write an accuracy specification for tensor operations that are verifiable**
   1. Write 2 or 3 (more or less approximated) specifications with pen and paper for low dimensions
      - then extrapolate specifications w.r.t. a formal variable representing the tensor dimension
      - and finally prove by induction that the specification is consistent for the component
   2. or generate the specification with symbolic formula of slide 4
      - using some “widening” operation (Abstract Interpretation wording) over the tensor dimension

2. Check if an implementation verifies a specification with analyses ensuring some guarantees
   - Check with stochastic arithmetic on some representative use-cases
   - Check with affine arithmetic and generate accuracy bounds on some representative numerical scenarios = use-cases with input ranges
     - Be ready to subdivide non-linear components
   - Check with symbolic arithmetic and generate accuracy formula
     - Be ready to guide the simplifications of the symbolic abstract terms

3. Show that an implementation is better than another one from the accuracy point of view
   - Compare the accuracy bounds/formula generated by the analyses

Symbolic arithmetic and affine arithmetic

- Symbolic arithmetic
  - $(x_0 + e_0) +_{\text{impl}} (x_1 + e_1) = (x_0 + x_1) + (\boldsymbol{e_0} + \boldsymbol{e_1}) + \big((x_0 + x_1) + (e_0 + e_1)\big) \times [-u, u]$
  - $(x_0 + e_0) *_{\text{impl}} (x_1 + e_1) = (x_0 \times x_1) + \big(\boldsymbol{e_0} \times \boldsymbol{x_1} + \boldsymbol{e_1} \times (\boldsymbol{x_0} + \boldsymbol{e_0})\big)$
    $+ \Big((x_0 \times x_1) + \big(e_0 \times x_1 + e_1 \times (x_0 + e_0)\big)\Big) \times [-u, u]$
  - $(x_0 + e_0) /_{\text{impl}} (x_1 + e_1) = \frac{x_0}{x_1} + \frac{\boldsymbol{x_0} \times \boldsymbol{e_1} - \boldsymbol{e_0} \times \boldsymbol{x_1}}{\boldsymbol{x_1} \times (\boldsymbol{x_1} + \boldsymbol{e_1})} + \left(\frac{x_0}{x_1} + \frac{x_0 \times e_1 - e_0 \times x_1}{x_1 \times (x_1 + e_1)}\right) \times [-u, u]$



  - The combination of operation $\Rightarrow$ big terms
  - No quantitative results
  - Exact formula, no over-approximation
  - Need some interval abstractions and simplifications to find generic formula

- Affine arithmetic
  - Normalization for non-linear operations $\Rightarrow$ some over-approximations (often too imprecise for large input intervals)
  - Quantitative results and intervals for the error
  - Limitation of the number of shared symbols to find generic formula

Specificities of the arithmetic with affine forms

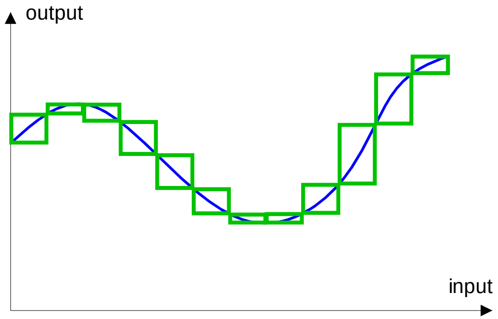
- The ideal value and the error (x, e) are represented by $\alpha_0 + \alpha_1 \times \varepsilon_1 + \alpha_2 \times \varepsilon_2 + \alpha_3 \times \varepsilon_3 + \ \dots$ where
  - $(\alpha_i)_{i\in[0,n]}$ are concrete values $\in \Re^{n+1}$, $(\varepsilon_i)_{i\in[1,n]}$ are symbolic shared variables $\in [-1, 1]^n$
  - Hence, $x \in [1.0, 6.0]$ becomes $x = 3.5 + 2.5 \times \varepsilon_1$, with $\varepsilon_1 = \frac{x-3.5}{2.5} \in [-1,1]$
- $(x_0 + e_0) *_{\text{impl}} (x_1 + e_1) = (x_0 \times x_1) + \left(\boldsymbol{e_0 \times x_1 + e_1 \times (x_0 + e_0)}\right)$

  $+$ introduced error of $\left((x_0 \times x_1) + \left(e_0 \times x_1 + e_1 \times (x_0 + e_0)\right)\right)$

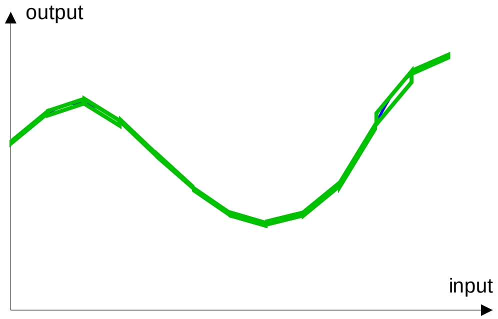
$(\alpha_0 + \alpha_1 \times \varepsilon_1 + \alpha_2 \times \varepsilon_2) + (\beta_0 + \beta_1 \times \varepsilon_1 + \beta_2 \times \varepsilon_2) = (\alpha_0 + \beta_0) + (\alpha_1 + \beta_1) \times \varepsilon_1 + (\alpha_2 + \beta_2) \times \varepsilon_2$, no additional over-approximation

$(\alpha_0 + \alpha_1 \times \varepsilon_1 + \alpha_2 \times \varepsilon_2) \times (\beta_0 + \beta_1 \times \varepsilon_1 + \beta_2 \times \varepsilon_2) = ((\alpha_0 \times \beta_0) + \frac{\alpha_1 \times \beta_1}{2} + \frac{\alpha_2 \times \beta_2}{2}) + (\alpha_1 \times \beta_0 + \beta_1 \times \alpha_0) \times \varepsilon_1 + (\alpha_2 \times \beta_0 + \beta_2 \times \alpha_0) \times \varepsilon_2$
$+ \frac{\alpha_1 \times \beta_1}{2} \times \mu_1 + \frac{\alpha_2 \times \beta_2}{2} \times \mu_2 + (\alpha_1 \times \beta_2 + \alpha_2 \times \beta_1)\mu_{12}$

  - over-approximation: $\mu_i = 2 \times \varepsilon_i^2 - 1$ is a new fresh symbolic variable $\in [-1, 1]$
  - over-approximation: $\mu_{ij} = \varepsilon_i \times \varepsilon_j$ is a new fresh symbolic variable $\in [-1, 1]$



- The simplification under the "normal" affine form is automatic
- The range of $\alpha_0 + \sum_{i=1}^{n} \alpha_i \times \varepsilon_i$ is $[\alpha_0 - \sum_{i=1}^{n} |\alpha_i|, \alpha_0 + \sum_{i=1}^{n} |\alpha_i|]$
- Non-linear operations generate over-approximations
  $\Rightarrow$ subdividing the input ranges improves the quality of the results



Example of Matrix multiplication

- Implicit need of a model of matrix multiplication with a numerical error containing all possible implementations
- Symbolic abstract inference of the error for this model with the rules of slide 4

Let us denote $|X| = \max_{i,j}|X_{ij}|$ and $|E_X| = \max_{i,j}|E_{xij}|$

- case $dim_X = (1,1),\ dim_Y = (1,1),\ (X) = (x_0 + e_{x0}), (Y) = \left(y_0 + e_{y0}\right)$
  - Symbolic induced error: $e_{y0} \times x_0 + e_{x0} \times \left(y_0 + e_{y0}\right)$
  - Approximate induced error:
    - $[-|X|, |X|] \times e_{y0} + [-|Y|, |Y|] \times e_{x0} + e_{x0} \times e_{y0}$
    - $[-|X| \times |E_Y| - (|Y| + |E_Y|) \times |E_X|, |X| \times |E_Y| + (|Y| + |E_Y|) \times |E_X|]$
  - Introduced error (*double*, and $|X|, |Y|$ normal floating-point):
    - $[-u, u] \times \left((x_0 + e_{x0}) \times \left(y_0 + e_{y0}\right)\right)$
    - $[-u, u] \times (|X| + |E_X|) \times (|Y| + |E_Y|)$

Example of Matrix multiplication (2)

Let us denote $|X| = \max_{i,j}|X_{ij}|$ and $|E_X| = \max_{i,j}|E_{xij}|$

- case $dim_X = (1,2),\ dim_Y = (2,1), (X) = (x_0 + e_{x0}, x_1 + e_{x1}), (Y) = \begin{pmatrix} y_0 + e_{y0} \\ y_1 + e_{y1} \end{pmatrix}$
  - Symbolic induced error: $e_{y0} \times x_0 + e_{x0} \times \left(y_0 + e_{y0}\right) + e_{y1} \times x_1 + e_{x1} \times \left(y_1 + e_{y1}\right)$
  - Approximate induced error:
    - $[-|X|, |X|] \times \left(e_{y0} + e_{y1}\right) + [-|Y|, |Y|] \times (e_{x0} + e_{x1}) + e_{x0} \times e_{y0} + e_{x1} \times e_{y1}$
    - $2 \times [-|X| \times |E_Y| - (|Y| + |E_Y|) \times |E_X|, |X| \times |E_Y| + (|Y| + |E_Y|) \times |E_X|]$
  - Introduced error:
    - $\left((1 + [-u, u]) \times \left((x_0 + e_{x0}) \times \left(y_0 + e_{y0}\right)\right) + (1 + [-u, u]) \times \left((x_1 + e_{x1}) \times \left(y_1 + e_{y1}\right)\right)\right) \times (1 + [-u, u])$
      $- \left((x_0 + e_{x0}) \times \left(y_0 + e_{y0}\right) + (x_1 + e_{x1}) \times \left(y_1 + e_{y1}\right)\right)$
      $= \left((x_0 + e_{x0}) \times \left(y_0 + e_{y0}\right) + (x_1 + e_{x1}) \times \left(y_1 + e_{y1}\right)\right) \times \left((1 + [-u, u])^2 - 1\right)$
    - $(2 + [-u, u]) \times [-u, u] \times (|X| + |E_X|) \times (|Y| + |E_Y|)$

Example of Matrix multiplication (3)

Let us denote $|X| = \max_{i,j}|X_{ij}|$ and $|E_X| = \max_{i,j}|E_{xij}|$

- case $dim_X = (1,3),\ dim_Y = (3,1), (X) = (x_0 + e_{x0}, x_1 + e_{x1}, x_2 + e_{x2}), (Y) = \begin{pmatrix} y_0 + e_{y0} \\ y_1 + e_{y1} \\ y_2 + e_{y2} \end{pmatrix}$
  - Symbolic induced error: $\sum_{i=0}^{2}\left(e_{yi} \times x_i + e_{xi} \times \left(y_i + e_{yi}\right)\right)$
  - Approximate induced error:
    - $\sum_{i=0}^{2}(e_{yi} \times [-|X|, |X|]) + \sum_{i=0}^{2}(e_{xi} \times [-|Y|, |Y|]) + \sum_{i=0}^{2}(e_{xi} \times e_{yi})$
    - $3 \times [-|X| \times |E_Y| - (|Y| + |E_Y|) \times |E_X|, |X| \times |E_Y| + (|Y| + |E_Y|) \times |E_X|]$
  - Introduced error:
    - $\left(\sum_{i=0}^{2}\left((x_i + e_{xi}) \times \left(y_i + e_{yi}\right)\right) \times \left((1 + [-u, u])^{4-i} - 1\right)\right)$ implementation associativity dependency
    - $\left[n - (1+u)^2 \times \frac{(1+u)^3 - 1}{u}, (1+u)^2 \times \frac{(1+u)^3 - 1}{u} - n\right] \times (|X| + |E_X|) \times (|Y| + |E_Y|)$
      less precise formula, but independent from any associativity rule

Example of Matrix multiplication (3) - extrapolation

- case $dim_X = (1, n), \; dim_Y = (n, 1)$
  - Symbolic induced error: $\sum_{i=0}^{n-1} \left(e_{yi} \times x_i + e_{xi} \times \left(y_i + e_{yi} \right) \right)$
  - Approximate induced error:
    - $\sum_{i=0}^{n-1}(e_{yi} \times [-|X|, |X|]) + \sum_{i=0}^{n-1}(e_{xi} \times [-|Y|, |Y|]) + \sum_{i=0}^{n-1}(e_{xi} \times e_{yi})$
    - $n \times [-|X| \times |E_Y| - (|Y| + |E_Y|) \times |E_X|, |X| \times |E_Y| + (|Y| + |E_Y|) \times |E_X|]$
  - Introduced error:
    - $\left(\sum_{i=0}^{n-1} \left((x_i + e_{xi}) \times \left(y_i + e_{yi} \right) \right) \times \left((1 + [-u, u])^{n+1-i} - 1 \right) \right)$
    - $\left[n - (1+u)^2 \times \frac{(1+u)^n - 1}{u}, (1+u)^2 \times \frac{(1+u)^n - 1}{u} - n \right] \times (|X| + |E_X|) \times (|Y| + |E_Y|)$
- Provable by induction over n – if your model of matrix multiplication is inductively defined over n

Example of Matrix multiplication (3) - generalization

- case $dim_X = (p,n),\ dim_Y = (n,q)$
  - Symbolic induced error for z_{ij}: $\sum_{k=0}^{n-1}\left(e_{ykj} \times x_{ik} + e_{xik} \times \left(y_{kj} + e_{ykj}\right)\right)$
  - Approximate induced error:
    - $\boldsymbol{n} \times [-|\boldsymbol{X}| \times |\boldsymbol{E_Y}| - (|\boldsymbol{Y}| + |\boldsymbol{E_Y}|) \times |\boldsymbol{E_X}|, |\boldsymbol{X}| \times |\boldsymbol{E_Y}| + (|\boldsymbol{Y}| + |\boldsymbol{E_Y}|) \times |\boldsymbol{E_X}|]$
    - $\sum_{k=0}^{n-1}(e_{ykj} \times [-|X|, |X|]) + \sum_{k=0}^{n-1}(e_{xik} \times [-|Y|, |Y|]) + \sum_{k=0}^{n-1}(e_{xik} \times e_{ykj})$

  - Introduced error:
    - $\left(\sum_{k=0}^{n-1}\left((x_{ik} + e_{xik}) \times \left(y_{kj} + e_{ykj}\right)\right) \times \left((1 + [-u, u])^{n+1-i} - 1\right)\right)$
    - $\left[\boldsymbol{n} - (\boldsymbol{1} + \boldsymbol{u})^2 \times \frac{(\boldsymbol{1}+\boldsymbol{u})^{\boldsymbol{n}} - \boldsymbol{1}}{\boldsymbol{u}}, (\boldsymbol{1} + \boldsymbol{u})^2 \times \frac{(\boldsymbol{1}+\boldsymbol{u})^{\boldsymbol{n}} - \boldsymbol{1}}{\boldsymbol{u}} - \boldsymbol{n}\right] \times (|\boldsymbol{X}| + |\boldsymbol{E_X}|) \times (|\boldsymbol{Y}| + |\boldsymbol{E_Y}|)$

- Provable by induction over p, q

Checking an implementation

- Ideally, use a symbolic abstract static analysis that has
  - simplifications over symbolic terms
  - a widening operator able to extrapolate formula and to prove them by induction

- If it uses double ($u' = 2^{-53}$) instead of float ($u = 2^{-24}$), it should prove
  - $\forall e_{ykj}, e_{xik}. \left|\sum_{k=0}^{n-1}\left(e_{ykj} \times [-|X|, |X|]\right) + \sum_{k=0}^{n-1}\left(e_{xik} \times [-|Y|, |Y|]\right) + \sum_{k=0}^{n-1}\left(e_{xik} \times e_{ykj}\right)\right|$
  $$\leq \boldsymbol{n \times [|X| \times |E_Y| + (|Y| + |E_Y|) \times |E_X|]}$$
  - $\forall e_{ykj}, e_{xik}. \left|\sum_{k=0}^{n-1}\left((x_{ik} + e_{xik}) \times \left(y_{kj} + e_{ykj}\right)\right) \times \left((1 + [-u', u'])^{n+1-i} - 1\right)\right|$
  $$\leq \left(\boldsymbol{(1+u)^2 \times \frac{(1+u)^n - 1}{u} - n}\right) \boldsymbol{\times (|X| + |E_X|) \times (|Y| + |E_Y|)}$$

  or $\boldsymbol{(1 + u')^2 \times \frac{(1+u')^n - 1}{u'}} \leq \boldsymbol{(1+u)^2 \times \frac{(1+u)^n - 1}{u}}$ for any n

Checking an implementation – more automatic

- Define a numerical scenario:
  - all coefficients are in [-1, 1],
  - choose $n = 6$
  - the absolute numerical error of the input coefficients is bound by 10^{-6}

- The static analysis (with FLDLib for instance) should be able to prove in `double` that the final error e
  - $|e| \leq 1.2 \times 10^{-5}$ (induced error) + $\left(\left(\mathbf{1} + \mathbf{2}^{-\mathbf{53}}\right)^{\mathbf{2}} \times \frac{(\mathbf{1}+\mathbf{2}^{-\mathbf{53}})^{\mathbf{6}}-\mathbf{1}}{\mathbf{2}^{-\mathbf{53}}} - \mathbf{6}\right) \times \left(\mathbf{1} + \mathbf{10}^{-\mathbf{6}}\right)^{\mathbf{2}}$ (introduced error)
  - no subdivision is required since the worst case has maximal coefficients for both matrices

From the implementation point of view

- When it computes $\sum_{k=0}^{n-1} x_{ik} \times y_{kj}$, it may compute $\left(\sum_{k=0}^{(n-1)/2} x_{ik} \times y_{kj}\right) + \left(\sum_{k=(n+1)/2}^{n-1} x_{ik} \times y_{kj}\right)$

- Hence for $n = 4$, it can claim that the introduced error is bound by

  $$4 \times \left((1+u)^3 - 1\right) \sim 7.153 \times 10^{-7} \text{ for } u = 2^{-24} \text{ the float semi-ulp}$$

  which is better than the specification

  $$(1+u)^2 \times \frac{(1+u)^4 - 1}{u} - 4 \sim 8.35 \times 10^{-7} \text{ for } u = 2^{-24} \text{ the float semi-ulp}$$

- It can also compare to another implementation

Work in Progress

- Writing models of SONNX components to be able to perform proof by induction

- Writing formula for SONNX components – ex convolution (model conv2d.c)
  - for the induced numerical error
  - for the introduced numerical error

- Writing symbolic abstract domains to synthesize accuracy formula with widening operator implementing proof by induction
  - Simplification of symbolic expressions
  - Extrapolation
  - Proof by induction

- Any help is welcome !

Conclusion – no silver bullet

- The accuracy formula will be approximated, so several specifications will be proposed depending on the approximation level

- The accuracy formula will be complex
  - but it is possible to show how they are obtained with the different steps of the methodology
  - these steps use induction over tensor dimension, so the more complex formula may be inductively defined w.r.t this dimension – we need a program to check them

- An implementation can check the specification
  - with test-cases in stochastic arithmetic
    - precise, fully automatic, realistic – standard deviation, but low guarantees
  - with numerical scenario in affine arithmetic
    - some formal guarantee, less precise and it may need annotations to subdivide
  - in symbolic arithmetic
    - strong guarantee, but induction proofs to fully automatize and certificate to produce (in why3?)
  - If the implementation use EFT to gain accuracy, it should explain it to the analysis

Thanks!