fn make_sized_bounded_int_checked_sum

Bridget Ma

Proves soundness of make_sized_bounded_int_checked_sum.

make_sized_bounded_int_checked_sum returns a Transformation that computes the sum of bounded ints. The effective range is reduced, as (bounds * size) must not overflow.

1 Hoare Triple

Precondition

- T (atomic input type and output type) is a type with trait Integer. Integer implies T has the trait bound:
  - CheckNull so that T is a valid atomic type for AtomDomain
  - CheckAtom to satisfy the preconditions of new_closed
  - for<'a> std::iter::Sum<&'a Self> so that the input vector can be summed
  - InfCast for casting a dataset distance of type usize to T
  - InfMul for multiplying a dataset distance of type T by a range of type T
  - InfSub so that the output domain is compatible with the output metric and for subtracting an upper bound of type T by a lower bound of type T

Pseudocode

```
def make_sized_bounded_int_checked_sum(
    size: usize,
    bounds: (T, T)
):
    input_domain: VectorDomain[AtomDomain[T].new_closed(bounds)].with_size(size), #
    input_metric: SymmetricDistance, #
    output_domain = AtomDomain(T) #

    if can_int_sum_overflow(size, bounds): #
        raise MakeTransformation("potential for overflow when computing function. " \
        "You could resolve this by choosing tighter clipping bounds or by using a " \
        "data type with greater bit-depth.")
    (lower, upper) = bounds.clone() #
    range = upper.inf_sub(lower) #

    def function(arg: Vec[T]) -> T: #
        return arg.iter().sum() #

    output_metric = AbsoluteDistance(T)

    def stability_map(d_in: IntDistance) -> AbsoluteDistance<T>:
        substitutions = T.inf_cast(d_in / 2) #
        return substitutions.inf_mul(range) #

```

```
25     return Transformation(
26         input_domain, output_domain, function,
27         input_metric, output_metric, stability_map)
```

Postcondition

Theorem 1.1. For every setting of the input parameters `(size, bounds)` to `make_sized_bounded_int_checked_sum` such that the given preconditions hold, `make_sized_bounded_int_checked_sum` raises an exception (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

1. (Appropriate output domain). For every element x in `input_domain`, `function`(x) is in `output_domain` or raises a data-independent runtime exception.

2. (Stability guarantee). For every pair of elements x, x' in `input_domain` and for every pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_metric`, if x, x' are `d_in`-close under `input_metric`, `stability_map`(`d_in`) does not raise an exception, and `stability_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_metric`.

2 Proofs

Proof. **(Part 1 – appropriate output domain).** The `output_domain` is `AtomDomain(T)`, so it is sufficient to show that `function` always returns non-null values of type `T`. By the definition of the `Sum` trait, `Iterator::sum` always returns a non-null value of type `T`. Thus, in all cases, the function (from line 16) returns a non-null value of type `T`. $\square$

Before proceeding with proving the validity of the stability map, we provide a couple lemmas.

Lemma 2.1. For vector u, v with each element $\ell \in u$, $z \in v$ drawn from domain $\mathcal{X}$ and `len(u)` $=$ `len(v)` $=$ `size`, denote U and V as the multisets of the elements u and v respectively. Let $A = U \cap V$. $|$`sum(u)` $-$ `sum(v)`$| = |\sum_{z \in U \setminus A} z - \sum_{z \in V \setminus A} z|$, where `sum` is an alias for `Iterator::sum`.

Proof. We have that

$$\begin{aligned} |\sum_{z \in U} z - \sum_{z \in V} z| &= |(\sum_{z \in U \setminus A} z + \sum_{z \in A} z) - (\sum_{z \in V \setminus A} z + \sum_{z \in A} z)| && \text{by properties of sets} \\ &= |\sum_{z \in U \setminus A} z - \sum_{z \in V \setminus A} z| && \text{by algebra} \end{aligned}$$

Conditioned on the correctness of the implementation of `Iterator::sum`, the variable `arg.iter().sum()` contains the sum of elements in `arg`. Therefore, $\sum_{z \in U} z$ and $\sum_{z \in V} z$ is equivalent to `sum(v)` and `sum(u)` respectively.

$$|\sum_{z \in U} z - \sum_{z \in V} z| = |\texttt{sum(u)} - \texttt{sum(v)}| = |\sum_{z \in U \setminus A} z - \sum_{z \in V \setminus A} z|$$

$\square$

Lemma 2.2. For vector u, v with each element $\ell \in u$, $z \in v$ drawn from domain $\mathcal{X}$ and `len(u)` $=$ `len(v)` $=$ `size`, $|$`function`$(u) -$ `function`$(v)| \leq \frac{d_{Sym}(u,v)}{2} \cdot$ `range`

Proof. Using the same notation as in 2.1, given `len(u)` $=$ `len(v)` $=$ `size`, by the definition of `ChangeOneDistance`, $|U \setminus A| = |V \setminus A| = d_{CO}(u, v) = d_{Sym}(u, v)/2$.
By pseudocode line 17, `function`$(u) =$ `sum`(u). Let $\alpha = \max_{(x,y) \in (U \setminus A \times V \setminus A) |x-y|}$. Given that every $\ell \in u$

must satisfy $\texttt{lower} \le \ell \le \texttt{upper}$, $\alpha \le \texttt{upper} - \texttt{lower} = \texttt{range}$, where $\texttt{upper} - \texttt{lower} = \texttt{range}$ by pseudocode line 14.

$$\begin{aligned}
|\texttt{function}(u) - \texttt{function}(v)| &= |\texttt{sum(u)} - \texttt{sum(v)}| \\
&= |\sum_{z \in U \setminus A} z - \sum_{z \in V \setminus A} z| && \text{by 2.1} \\
&\le |U \setminus A| \cdot \alpha && \text{by algebra} \\
&= \frac{d_{Sym}(u, v)}{2} \cdot \alpha \\
&\le \frac{d_{Sym}(u, v)}{2} \cdot \texttt{range}
\end{aligned}$$

□

Proof. **(Part 2 – stability map).** Take any two elements u, v in the `input_domain` and any pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_metric`. Assume u, v are `d_in`-close under `input_metric` and that `stability_map`(`d_in`) $\le$ `d_out`. These assumptions are used to establish the following inequality:

$$\begin{aligned}
|\texttt{function}(u) - \texttt{function}(v)| &\le \frac{d_{Sym}(u, v)}{2} \cdot \texttt{range} && \text{by 2.2} \\
&\le \frac{\texttt{d_in}}{2} \cdot \texttt{range} && \text{by } \texttt{InfCast} \\
&\le \texttt{T.inf_cast(d_in/2).inf_mul(range)} && \text{by } \texttt{InfMul} \\
&= \texttt{stability_map(d_in)} && \text{by pseudocode line 23} \\
&\le \texttt{d_out} && \text{by the second assumption}
\end{aligned}$$

It is shown that `function`(u), `function`(v) are `d_out`-close under `output_metric`. □