'

Ideas for optimizing generic operations

week of 8 July

Case 1:

Given two generic operations A and B whose indexing map for ins and outs arguments is the id map. If:

- $out(A) == in(B)$

Combine both operations into one by combining their inner blocks.

Example

$$\begin{cases} A = generic(id, id, parallel, ins(X)\ outs(Y)\{y = f(x)\}) \\ B = generic(id, id, parallel, ins(Y)\ outs(Z)\{z = g(y)\}) \end{cases} \\ \implies \\ C = generic(id, id, parallel, ins(X)\ outs(Z)\{z = g\left(f(x)\right)\}) \tag{1}$$

Ways to express generic operations:

Indexing maps are id:

Let's consider the following generic op where $len(A) = N$:

$$\text{A} = \text{generic(id, id, parallel, ins(X) outs(Y)\{block(x,y): y = f(x)\})} \quad (2)$$

We can see the generic operation, in this case, as a build operation which produces a vector of size N applying the function f to each element $X(i)$. Hence, we get the following:

$$A = build \quad N \quad \left(i \rightarrow f\left(X\left[i\right]\right)\right) \qquad (3)$$

Ways to express generic operations:

Indexing maps are id:

Rewrite rules for equality saturation: [1]

- generic(id, id, parallel, ins(X) outs(Y) {y = f(x)}) = build N $(\mathrm{i} \to f\,(X\,[i]))$
- $(build\;\; N\;\; f\,)\,[i] = f\,(i\,)$
- $(x \to f\,(x))\;\; i = f\,(i\,)$
- build N $(\mathrm{i} \to f\,(X\,[i])) = generic(id, id, parallel, ins(X)\;\, outs(Y)\{y = f\,(x)\})$.
  Y can be given any unique variable name, it only represents the name of the output.

[1] some rules are inspired from Jonathan's paper

Ways to express generic operations:

Indexing maps are id:

Let us consider the earlier example to see how these rules derive the following result for us:

$$
\begin{aligned}
&\begin{cases} generic(id, id, parallel, ins(X)\ outs(Y)\{y = f(x)\}) \\ generic(id, id, parallel, ins(Y)\ outs(Z)\{z = g(y)\}) \end{cases} \\
&= build\ N\left(j \to g\left[\left(build\ N\ \left(i \to f\left(X[i]\right)\right)\right)[j]\right]\right) \\
&= build\ N\left(j \to g\left[\left(i \to f\left(X[i]\right)\right)[j]\right]\right) \\
&= build\ N\left(j \to g\left[f\left(X[j]\right)\right]\right) \\
&= generic(id, id, parallel, ins(X)\ outs(Z)\{z = g\left(f(x)\right)\})
\end{aligned}
\tag{4}
$$

Ways to express generic operations:

Arbitrary indexing maps:

Let us examine the next situation, where the input arguments have more intricate indexing maps. The output indexing maps are kept as id maps because we want to assign a value to every element in the resulting array. The rewrite rules remain mostly unchanged:

- $\text{generic}\Big((i \to h(i))\,, id, parallel, ins(X)\ outs(Y)\{y = f(x)\}\Big) = build\ \ N\ \ \Big(i \to f\,\Big(X\,\big[h(i)\big]\Big)\Big)$
- $(build\ \ N\ \ f)\,\big[h(i)\big] = f[h(i)]$
- $(x \to f(x))\ h(i) = f(h(i))$ $\quad$ *substitute* x *by* $h(i)$
- $\text{build}\ \ \text{N}\ \ (\text{i} \to f\,\Big(X\,\big[h(i)\big]\Big)) = generic\,\Big((i \to h(i))\,, id, parallel, ins(X)\ outs(Y)\{y = f(x)\}\Big)$.

Ways to express generic operations:

Arbitrary indexing maps:

Let us consider the following example to see how these rules derive the following result for us:

$$
\begin{cases}
generic((i \to h(i)), id, parallel, ins(X)\ outs(Y)\{y = f(x)\}) \\
generic((j \to l(j)), id, parallel, ins(Y)\ outs(Z)\{z = g(y)\})
\end{cases}
$$

$$
\begin{aligned}
&= build\ \ N \left(j \to g \left[\left(build\ \ N\ \ \left(i \to f \left(X \left[h(i) \right] \right) \right) \right) \left[l(j) \right] \right] \right) \\
&= build\ \ N \left(j \to g \left[\left(i \to f \left(X \left[h(i) \right] \right) \right) \left[l(j) \right] \right] \right) \\
&= build\ \ N \left(j \to g \left[f \left(X \left[h \left(l \left(j \right) \right) \right] \right) \right] \right) \\
&= generic(i \to h\left(l \left(i \right) \right), id, parallel, ins(X)\ outs(Z)\{z = g\left(f(x) \right) \}) \qquad (5)
\end{aligned}
$$