Simplifying `vec` expressions*

Kevin S. Van Horn
Adobe Inc.

March 29, 2019

1 Definition of `vec`

In the following, $\mathbb{R}$ is the set of scalars (real numbers), $\mathbb{R}^*$ is the set of vectors over $\mathbb{R}$, $(v \circ w)$ is the vector obtained by appending vectors v and w, and $(s\colon v)$ is the vector obtained by prepending scalar s to vector v.

We define the function vec as follows:

1. The domain of vec is all n-tuples $(x_1, \ldots, x_n)$, $n \geq 0$, such that $x_i \in \mathbb{R}$ or $x_i \in \mathbb{R}^*$ for all $1 \leq i \leq n$. The range is $\mathbb{R}^*$.

2. If $n \geq 0$, $s \in \mathbb{R}$, and $v \in \mathbb{R}^*$, then

$$\begin{aligned} \operatorname{vec}() &= \text{the length-0 empty vector} \\ \operatorname{vec}(s, x_1, \ldots, x_n) &= s\colon \operatorname{vec}(x_1, \ldots, x_n) \\ \operatorname{vec}(v, x_1, \ldots, x_n) &= v \circ \operatorname{vec}(x_1, \ldots, x_n) \end{aligned}$$

2 Normal form

Definition. An expression v is said to be *veckish* if it has the form $v = \operatorname{vec}(e_1, \ldots, e_n)$.

Definition. An expression $\operatorname{vec}(e_1, \ldots, e_n)$ is *nullary* if $n = 0$.

Definition. An expression $\operatorname{vec}(e_1, \ldots, e_n)$ is *element-normal* if $n > 0$ and $e_i \in \mathbb{R}$ for all $1 \leq i \leq n$.

Definition. An expression $\operatorname{vec}(e_1, \ldots, e_n)$ is *quasi-append-normal* if

- all $e_i \in \mathbb{R}^*$,
- any veckish e_i is element-normal, and

- no two consecutive e_i are veckish.

It is *append-normal* if it is both quasi-append-normal and $n > 1$.

Definition. A veckish expression is in *normal form* if it is either nullary, element-normal, or append-normal.

3 Quasi-normal triples

The simplification algorithm for veckish expressions is based on the idea of a *quasi-normal triple*, which is used to scan through the arguments of a veckish expression to build a new, equivalent expression that is in normal form.

Definition. An expression $\operatorname{vec}(x_1, \ldots, x_n)$ is *scalar-initial* if $n \geq 1$ and x_1 is either a scalar expression or a scalar-initial veckish expression.

Definition. An expression $\operatorname{vec}(x_1, \ldots, x_n)$ is *scalar-final* if $n \geq 1$ and x_n is either a scalar expression or a scalar-final veckish expression.

Proposition. *If $n > 0$ and $u = \operatorname{vec}(x_1, \ldots, x_n)$ is quasi-append-normal then u is scalar-final iff x_n is element-normal.*

Definition. A *quasi-normal triple* is a triple of veckish expressions (u, v, w) such that

- u is quasi-append-normal,
- v is nullary or element-normal,
- if u is scalar-final then v is nullary and w is not scalar-initial.

You can think of u as the result arguments we have already produced, v as an element-normal result argument we are in the process of constructing, and w as the arguments we have not yet processed.

Definition. Two quasi-normal triples $a = (u, v, w)$ and $b = (u', v', w')$ are *equivalent*, written $a \equiv b$, if

$$u \circ v \circ w = u' \circ v' \circ w'.$$

Similarly, quasi-normal triple (u, v, w) is equivalent to expression e if $e = u \circ v \circ w$.

In order to show that our algorithm terminates, we'll need to show that some "size" metric decreases at each iteration.

Definition. The *vec-size* of an expression is defined by

$$\operatorname{vsize}(e) = 1 \quad \text{if } e \text{ is not veckish}$$

$$\operatorname{vsize}(\operatorname{vec}(x_1, \ldots, x_n)) = 1 + \sum_{i=1}^{n} \operatorname{vsize}(x_1)$$

Definition. The *QNT-size* of a quasi-normal triple is defined by

$$\mathrm{qntsize}(u, v, w) = 2 \cdot \mathrm{vsize}(w) + \begin{cases} 0 & \text{if } v \text{ is nullary} \\ 1 & \text{otherwise} \end{cases}$$

Definition. $a \rightarrowtail b$ ("a reduces to b") means that if a is a quasi-normal triple, then

- b is a quasi-normal triple,
- $b \equiv a$, and
- $\mathrm{qntsize}(b) < \mathrm{qntsize}(a)$.

Our algorithm works by successively reducing a quasi-normal triple until no further reduction is possible. Here are the reductions we use.

Proposition 1. *The following reduction properties hold:*

1. *If s' is a scalar expression then*

$$\begin{gathered}(u,\ \mathrm{vec}\,(s_1, \ldots, s_m)\,,\ \mathrm{vec}\,(s', x_1, \ldots, x_n)) \rightarrowtail \\ (u, \mathrm{vec}\,(s_1, \ldots, s_m, s')\,, \mathrm{vec}\,(x_1, \ldots, x_n))\,.\end{gathered}$$

2.

$$\begin{gathered}(u,\ v,\ \mathrm{vec}\,(\mathrm{vec}\,(y_1, \ldots, y_k)\,, x_1, \ldots, x_n)) \rightarrowtail \\ (u,\ v,\ \mathrm{vec}\,(y_1, \ldots, y_k, x_1, \ldots, x_n))\,.\end{gathered}$$

3. *If $v$is not nullary and w does not have a scalar or veckish first argument, then*

$$\begin{gathered}(\mathrm{vec}\,(y_1, \ldots, y_m)\,,\ v,\ w) \rightarrowtail \\ (\mathrm{vec}\,(y_1, \ldots, y_m, v)\,,\ \mathrm{vec}(),\ w)\,.\end{gathered}$$

4. *If y is neither scalar nor veckish, then*

$$\begin{gathered}(\mathrm{vec}\,(v_1, \ldots, v_m)\,,\ \mathrm{vec}(),\ \mathrm{vec}\,(y, x_1, \ldots, x_n)) \rightarrowtail \\ (\mathrm{vec}\,(v_1, \ldots, v_m, y)\,,\ \mathrm{vec}(),\ \mathrm{vec}\,(x_1, \ldots, x_n))\,.\end{gathered}$$

We'll need to show that, on termination, we have a normal-form expression equivalent to the original. We'll use the following.

Proposition 2. *If (u, v, w) is a quasi-normal triple that does not match any of reductions 1–4, then both v and w are nullary, hence $(u, v, w) \equiv u$. Furthermore,*

- *if u has exactly one argument u', so that $u = \mathrm{vec}\,(u')$, then u' is a vector expression that either is not veckish or is in normal form;*

- *otherwise u is in normal form.*

Proof. As follows:

- w does not have a scalar first argument (no match with reduction 1).
- w does not have a veckish first argument (no match with reduction 2).
- v is nullary (no match with reduction 3, and w has no scalar or veckish first argument).
- w is nullary (no match with reduction 4, v is nullary, w has no scalar or veckish first argument).

If u is nullary, then it is by definition in normal form. If u has exactly one argument u', then since u is quasi-append-normal, u' is a vector expression that is either not veckish or is element-normal; in the latter case u' is by definition in normal form. If u has more than one argument then it is append-normal, and hence in normal form. □

4 Algorithm for simplifying veckish expressions

Here is the algorithm:

- Input: a veckish expression e, all of whose arguments have either scalar or vector type.
- Output: an expression e' equivalent to e which, if veckish, is in normal form.
- Pseudocode:

```
u := vec(); v := vec(); w := e;
while (u, v, w) matches any of reductions 1--4:
    apply a matching reduction;
if u has exactly one argument:
    return that argument;
else:
    return u;
```

From Proposition 1 we have that $(u, v, w) \equiv e$ is an invariant of the while loop, and $\text{qntsize}\,(u, v, w)$ decreases at each iteration. Since $\text{qntsize}\,(u, v, w)$ is by definition a positive integer, it cannot decrease indefinitely, so the loop terminates. Proposition 2 then tells us that the returned value is equivalent to e and, if veckish, is in normal form.