Application of the proof strategy

Let A stands for the formula

```
(and (integerp n)
     (integerp key)
     (integerp entr)
     (integer-listp a)
)
```

Conjuncts in A state that variables n, key, $entr$ are integers and a is an integer list.

Let B denotes formula

```
(and (< 0 n)
     (<= n (length a))
     (< 0 entr)
)
```

with an obvious meaning.

The formula C of the form

```
(<= entr (cnt 0 (- n 1) key a))
```

states that the number of occurrences of key within subsequence of a in between indices 0 and $n-1$ is not less than $entr$.

Another abbreviation D stands for

```
(= (rep2 n key entr a) 1)
```

meaning that the variable $result$ is equal to 1 just after execution of a loop which can be modeled by invocation of function $rep2$ with arguments n, key, $entr$ and a.

The formula E

```
(> entr (cnt 0 (- n 1) key a))
```

obviously is negation of C.

The formula F

```
(= (rep2 n key entr a) 0)
```

is a counterpart of D corresponding to unsuccessful search for key during loop execution.

The formula J

```
(= entr (rep1 n key entr a))
```

expresses equality of values of $entr$ and cnt just after execution of the loop. According to the symbolic verification method this execution is modeled by invocation of function $rep1$ with arguments n, key, $entr$ and a.

Finally, the formula K of the form

```
(zp n)
```

states that n is an integer greater than zero.

Now we aggregate these meta-formulas into bigger ones. Let $L \equiv A \wedge B \wedge C$ and $M \equiv A \wedge B \wedge E$. The initial underlying theory in this case is the definition of `cnt`, `rep1` and `rep2`.

Just before the step (1) of our algorithm the VC ϕ corresponds to the following pattern:

$$A \Rightarrow (B \Rightarrow ((C \Rightarrow D) \wedge (E \Rightarrow F))).$$

The step (1) transforms ϕ into ϕ_1 corresponding to the pattern

$$(L \Rightarrow D) \wedge (M \Rightarrow F).$$

Note that at this moment the equivalence still holds.

The step (2) does not change ϕ_1, so we proceed to step (3). It generates automatically a non-recursive definition for the function `rep2`. This new definition is equivalent to the original one since ACL2 was able to prove automatically by induction on `n` the following theorem:

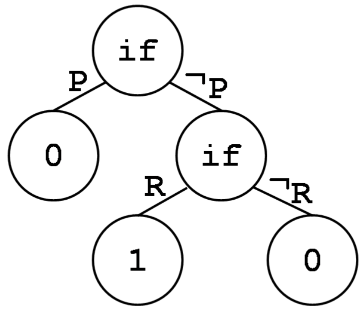
$$(A \wedge B \wedge \neg J) \Rightarrow F.$$

I asserts that whenever the loop exit condition J is false, the value of `rep2` is equal to the initial value of `result`, i.e. `0`. In the process of replacement operation generation we can determine that within the body of `rep2` the function `rep1` does not invoke any other function but itself. This guarantees automatically that `rep2` was redefined in a non-recursive way.

Let us consider such a non-recursive redefinition:

```
(defun rep2(i key entr a)
    (if
        (zp i)
        0
        (if
            (= entr (rep1 i key entr a))
            1
            0
        )
    )
)
```

It results in the following tree representation of `rep2`:



Let us denote it as $rep2\_tree$. The edge labeled by P corresponds to formula `(zp i)`, whereas R stands for `(= entr (rep1 i key entr a))`.

The repeating application of the step (1) takes into consideration the sub-formula D in ϕ_1 because it contains an invocation of `rep2`. The new representation $rep2\_tree_1$ results from $rep2\_tree$ after replacement of all variables in edge labels by corresponding arguments of `rep2` invocation in formula D. Actually, this replacement looks like $(i \leftarrow n, key \leftarrow key, entr \leftarrow entr, a \leftarrow a)$. Let labels P_1, $\neg P_1$, R_1 and $\neg R_1$ denote P, $\neg P$, R and $\neg R$ correspondingly after such substitution. If we recall our meta-formulas from the starting paragraphs then obviously P_1 is K and R_1 is J.

Now, replacing D in ϕ_1 by conjunction of specific implications provided by $rep2\_tree_1$ we construct the formula ϕ_2:

$$\begin{gathered}(L \Rightarrow \\ ((K \Rightarrow (0 = 1))\wedge \\ ((\neg K \wedge J) \Rightarrow (1 = 1))\wedge \\ ((\neg K \wedge \neg J) \Rightarrow (0 = 1))))\wedge \\ (M \Rightarrow F)\end{gathered}$$

What is the conclusion of every such implication? We merely replace $rep2$ in D with an expression defined by implication antecedent. The first implication corresponds to the path in $rep2\_tree_1$ traversing the edge P_1. By analogy, the second implication addresses the path over $\neg P_1$ and R_1. Finally, the third one relates to the path over $\neg P_1$ and $\neg R_1$.

Next, we consider sub-formula F in ϕ_1. By analogy with $rep2\_tree_1$ we build up $rep2\_tree_2$. It turns out that $rep2\_tree_2 = rep2\_tree_1$.

The tree $rep2\_tree_2$ provides us with another set of special implications. If we substitute their conjunction instead of F then we transform ϕ_2 into ϕ_3:

$$\begin{gathered}(L \Rightarrow \\ ((K \Rightarrow (0 = 1))\wedge \\ ((\neg K \wedge J) \Rightarrow (1 = 1))\wedge \\ ((\neg K \wedge \neg J) \Rightarrow (0 = 1))))\wedge \\ (M \Rightarrow \\ ((K \Rightarrow (0 = 0))\wedge \\ ((\neg K \wedge J) \Rightarrow (1 = 0))\wedge \\ ((\neg K \wedge \neg J) \Rightarrow (0 = 0))))\end{gathered}$$

The obvious logical rewritings (like de Morgan laws or $\neg\phi \vee \psi \equiv \phi \to \psi$) transform ϕ_3 into ϕ_4

$$\begin{gathered}(L \Rightarrow \\ ((\neg K \vee (0 = 1))\wedge \\ ((K \vee \neg J) \vee (1 = 1))\wedge \\ ((K \vee J) \vee (0 = 1))))\wedge \\ (M \Rightarrow \\ ((\neg K \vee (0 = 0))\wedge \\ ((K \vee \neg J) \vee (1 = 0))\wedge \\ ((K \vee J) \vee (0 = 0))))\end{gathered}$$

which in turn reforms into a more normalized ϕ':

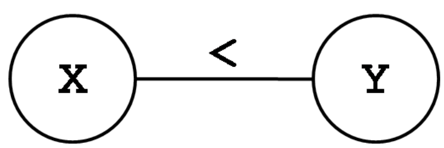
$$\begin{gathered}(L \Rightarrow (\neg K \vee (0 = 1)))\wedge \\ (L \Rightarrow ((K \vee \neg J) \vee (1 = 1)))\wedge \\ (L \Rightarrow ((K \vee J) \vee (0 = 1)))\wedge \\ (M \Rightarrow (\neg K \vee (0 = 0)))\wedge \\ (M \Rightarrow ((K \vee \neg J) \vee (1 = 0)))\wedge \\ (M \Rightarrow ((K \vee J) \vee (0 = 0)))\end{gathered}$$

Note that until this very step we preserve equivalence of our formulas.

Since ϕ has changed we can repeat steps (1)–(4). Exactly, the step (2) may transform the following disjunct of ϕ':

$$S \equiv (M \Rightarrow ((K \vee \neg J) \vee (1 = 0))).$$

The relation graph has been produced for S. Consider the following component of the graph:



The label X stands for $(cnt\ 0\ (-\ n\ 1)\ key\ a)$ whereas Y means $entr$. Remind that this graph component is actually formula E.

Which subgoal will lead to modification of ϕ'? In fact it is $\neg J$ corresponding to the pattern $g(c, d)$ where g is "$\neq$", c is $entr$ and d is $(rep1\ n\ key\ entr\ a)$. So, the searching procedure begins to look for a function call corresponding to the variable $entr$. The search begins at node X. During the search a subgraph of the relation graph emerges. This subgraph is exactly the component demonstrated above. The expression $(cnt\ 0\ (-\ n\ 1)\ key\ a)$ is the function call we were looking for. The conjunct E is the path we need. So, the expression $(cnt\ 0\ (-\ n\ 1)\ key\ a)$ must be assigned to variable v, and E becomes the value of q.

As a result we have the new formula

$$T \equiv (=\ (cnt\ 0\ (-\ n\ 1)\ key\ a)\ (rep1\ n\ key\ entr\ a)).$$

Let Z denote the disjunct S after replacement of the goal $\neg J$ by T:

$$(M \Rightarrow ((K \vee T) \vee (1 = 0))).$$

Note that $Z \not\Leftarrow S$ but the truth of S follows from the truth of Z. So we may replace S by Z in ϕ' which results in formula ϕ'':

$$\begin{gathered}(L \Rightarrow (\neg K \vee (0 = 1)))\wedge \\ (L \Rightarrow ((K \vee \neg J) \vee (1 = 1)))\wedge \\ (L \Rightarrow ((K \vee J) \vee (0 = 1)))\wedge \\ (M \Rightarrow (\neg K \vee (0 = 0)))\wedge \\ (M \Rightarrow ((K \vee T) \vee (1 = 0)))\wedge \\ (M \Rightarrow ((K \vee J) \vee (0 = 0)))\end{gathered}$$

And here it is, the result of our strategy. On the step (6) ACL2 is able to prove ϕ'' by induction on n thus validating the original VC ϕ.