From kaufmann Fri Sep  6 12:11:08 1991
To: windley@cs.uidaho.edu
Subject: example

Hi, Phil --

I went ahead and did that example. It took me just under an hour at the terminal; I'd already written out a page of definitions and lemmas, as indicated below (I think I showed them to you). The final theorem, main-theorem, says that there are arbitrarily large n such that running the interpreter for n steps, from the initial state, results in a return to the initial state. But the content of that theorem is really in main-lemma, which has no quantifiers and is where the proof effort lies.

Often in such efforts I’ll go back and get rid of the INSTRUCTIONS hints, which are printed out by the system when a proof using my interactive ‘proof-checker’ is completed. By ‘get rid of’ I mean that I’ll try to prove suitable rewrite rules so that I can produce an input file without INSTRUCTIONS hints, which can go through the unadorned Boyer-Moore prover. I haven’t done that here, simply because I can’t see that it’s worth the time (hard to tell how long it would take). So, I suspect that the ‘proof’ I finally got is quite similar to one you would obtain, except that the system helped me a lot with the arithmetic (even though I didn’t load in any libraries), inductions were easy to hint, and I could do lots of simplifications at the subterm level.

Anyhow, below is the input file, which takes about 53 seconds of real time to replay through pc-nqthm on a Sun 3/60.

-- Matt

EVENT: Start with the initial nqthm theory.

;;;; I had preliminary definitions of step, interp, and interp-prop written
;;;; out on paper when starting this, along with statements of the lemmas
;;;; interp-plus and main-lemma. I’ll time the rest....

DEFINITION:
step(s, d0)
= let state be car(s),
    d be cdr(s)
in case on state:
case = 0 then cons(1, d0)
case = 1 then if 0 < d then cons(2, d)
    else cons(3, d) endif
case = 2 then cons(1, d - 1)
otherwise cons(0, 0) endcase endlet

Definition:
interp(s, d0, n) = if n ≃ 0 then s
    else interp(step(s, d0), d0, n - 1) endif

Theorem: interp-plus
interp(s, d0, m + n) = interp(interp(s, d0, m), d0, n)

Theorem: plus-assoc
((x + y) + z) = (x + (y + z))

Theorem: interp-from-2
(w ≠ 0) → (interp(cons(2, d), d0, w) = interp(cons(1, d - 1), d0, w - 1))

Theorem: numberp-cdr-interp
((d0 ∈ N) ∧ (cdr(cons-state-d) ∈ N))
    → (cdr(interp(cons-state-d, d0, w)) ∈ N)

Theorem: through-the-loop
((d ∈ N) ∧ (d0 ∈ N))
    → (interp(cons(1, d), d0, d + d + w) = interp(’(1 . 0), d0, w))

Theorem: main-lemma
(d0 ∈ N)
    → (interp(’(0 . 0), d0, k *(3 + d0 + d0)) = ’(0 . 0))

# The following generates the four forms following it:
(defn-sk+ interp-prop (d0)
    (forall x (exists n (and (lessp x n)
        (equal (interp (cons 0 0) d0 n)
            (cons 0 0)))))
)
**Definition:**
interp-prop \((\text{d}0)\)
\[\iff \forall x \exists n ((x < n) \land (\text{interp}(\text{cons}(0, 0), \text{d}0, n) = \text{cons}(0, 0)))\]

**Event:** Disable interp-prop.

**Theorem:** interp-prop-suff
\[((x (\text{d}0) < n) \land (\text{interp}(^\_0 . 0, \text{d}0, n) = ^\_0 . 0))\]
\[\rightarrow \text{interp-prop}(\text{d}0)\]

**Theorem:** interp-prop-necc
\[(\neg ((x < n(\text{d}0, x)) \land (\text{interp}(^\_0 . 0, \text{d}0, n), n(\text{d}0, x)) = ^\_0 . 0)))\]
\[\rightarrow (\neg \text{interp-prop}(\text{d}0))\]

**Theorem:** main-theorem-hack-lemma
\((x < ((1 + x) * (3 + w))) = t\)

**Theorem:** main-theorem
\((\text{d}0 \in \text{N}) \rightarrow \text{interp-prop}(\text{d}0)\)
Index

exists, 3
forall, 3
interp, 2, 3
interp-from-2, 2
interp-plus, 2
interp-prop, 3
interp-prop-necc, 3
interp-prop-suff, 3

main-lemma, 2
main-theorem, 3
main-theorem-hack-lemma, 3

n, 3
numberp-cdr-interp, 2

plus-assoc, 2

step, 1, 2

through-the-loop, 2

x, 3