Stateman: Using Metafunctions to Manage Large Terms Representing Machine States

J Strother Moore
Department of Computer Science
University of Texas at Austin

Terms Representing Machine States

```
(!I 6                              ; set pc to 6
 (!S NIL                           ; set status flag to NIL
  (!R 0 8 4280                     ; set mem[0..7] to 4280
   (!R 16 8 (LOGAND x y)
    (!R (+ 40 (* 8 i)) 8 111
     (!R 4280 8 999
      ST))))))
```

Terms Representing Machine States

```
;              addr                n    val
 (!I                                    6
  (!S                                   NIL
   (!R    0                        8    4280
    (!R   16                       8    (LOGAND x y)
     (!R  (+ 40 (* 8 i))           8    111
      (!R 4280                     8    999
       ST))))))
```

Terms Representing Machine States

```
(I
 (!I                                    6
  (!S                                   NIL
   (!R     0                      8     4280
    (!R    16                     8     (LOGAND x y)
     (!R   (+ 40 (* 8 i))         8     111
      (!R 4280                    8     999
       ST)))))))
=
6
```

Terms Representing Machine States

```
(S
 (!I                                        6
  (!S                                       NIL
   (!R    0                           8     4280
    (!R   16                          8     (LOGAND x y)
     (!R  (+ 40 (* 8 i))              8     111
      (!R 4280                        8     999
       ST)))))))
=
NIL
```

Terms Representing Machine States

```
(R 16 8
 (!I                                  6
  (!S                                 NIL
   (!R    0                     8     4280
    (!R   16                    8     (LOGAND x y)
     (!R  (+ 40 (* 8 i))        8     111
      (!R 4280                  8     999
       ST)))))))
```

$=$

(MOD (LOGAND x y) 2^{64})

Terms Representing Machine States

```
(R 4280 8
 (!I                                  6
  (!S                                 NIL
   (!R    0                      8    4280
    (!R   16                     8    (LOGAND x y)
     (!R  (+ 40 (* 8 i))         8    111
      (!R 4280                   8    999
       ST)))))))
```

$=$

999

provided (+ 40 (* 8 i) 8) $\leq$ 4280
$\quad\quad\vee$ (+ 4280 8) $\leq$ (+ 40 (* 8 i))

Rewrite Rules

- $(\texttt{I}\ (\texttt{!S}\ v\ st)) = (\texttt{I}\ st)$

- $\big((\texttt{NATP}\ a) \wedge (\texttt{NATP}\ b) \wedge (\texttt{+}\ b\ k) \leq a\big)$
  $\to (\texttt{R}\ a\ n\ (\texttt{!R}\ b\ k\ v\ st)) = (\texttt{R}\ a\ n\ st)$

- $\big((\texttt{NATP}\ a) \wedge (\texttt{NATP}\ b) \wedge (\texttt{+}\ a\ n) \leq b\big)$
  $\to (\texttt{R}\ a\ n\ (\texttt{!R}\ b\ k\ v\ st)) = (\texttt{R}\ a\ n\ st)$

Such rules suffice to manipulate state expressions.

Rewrite Rules

- $\texttt{(I (!S}\ v\ st\texttt{))} = \texttt{(I}\ st\texttt{)}$

- $\big(\texttt{(NATP}\ a\texttt{)} \wedge \texttt{(NATP}\ b\texttt{)} \wedge \texttt{(+}\ b\ k\texttt{)} \leq a\big)$
  $\rightarrow \texttt{(R}\ a\ n\ \texttt{(!R}\ b\ k\ v\ st\texttt{))} = \texttt{(R}\ a\ n\ st\texttt{)}$

- $\big(\texttt{(NATP}\ a\texttt{)} \wedge \texttt{(NATP}\ b\texttt{)} \wedge \texttt{(+}\ a\ n\texttt{)} \leq b\big)$
  $\rightarrow \texttt{(R}\ a\ n\ \texttt{(!R}\ b\ k\ v\ st\texttt{))} = \texttt{(R}\ a\ n\ st\texttt{)}$

Such rules suffice to manipulate state expressions — except when there are deep nests of !R-expressions and a, b, n, and k are large expressions.

Terms Representing Machine States

```
(!I                                6
 (!S                               NIL
  (!R     0                   8    4280
   (!R    16                  8    (LOGAND x y)
    (!R   (+ 40 (* 8 i))      8    111
     (!R 4280                 8    999
      ST))))))
```

Size (in function applications):	9
Biggest Address or Value Expression:	2

Motivation for This Project

We have recently analyzed a piece of code (15,361 instructions of a formal ISA) involving states with:

Size (in function applications):	2,158,895
Biggest Address or Value Expression:	147,233

Backchaining to decide questions like

$$\texttt{(+ 40 (* 8 } i\texttt{) 8)} \leq \texttt{4280}$$
$$\quad \vee \texttt{ (+ 4280 8)} \leq \texttt{(+ 40 (* 8 } i\texttt{))}$$

for every pair of addresses in such state expressions is prohibitive.

Highlights

- Manage read-over-write and write-over-write expressions exclusively with metafunctions
- Implement a syntactic interval inference mechanism
- Implement syntactic means of deciding some inequalities
- Implement syntactic means of simplifying some `MOD` expressions
- Use syntactic means to decide overlap questions
- Insist that all byte counts be quoted constants

- Do not put nested `!R`-expressions into address order
- Eliminate perfectly shadowed writes
- Use `hons` rather than `cons` to create state expressions
- `HIDE` the state expressions produced by the metafunctions
- `HIDE` some values extracted by reads from hidden states to avoid re-simplifying them
- Prove guards and well-formedness guarantees of the metafunctions

Ainni — Our Interval Analyzer

Given

`(+ 288 (* 8 (LOGAND 31 (ASH (R 4520 8` st`) -3))))`

our analyzer reports an interval of [288, 536].

But if `(R 4520 8` st`)` < 24 is known by context, then the interval shrinks to [288, 304].

The analyzer can compute the interval $[0, 2^{32} - 1]$ for the largest value term encountered (147,233 function applications) in 0.01 seconds.

Examples of Ainni

(switch to *shell* buffer)

Finding Assignments

```
(R 4280 8
 (!I                                  6
  (!S                                 NIL
   (!R    0                     8     4280
    (!R   16                    8     (LOGAND x y)
     (!R  (+ 40 (* 8 i))        8     111
      (!R 4280                  8     999
       ST)))))))
```

=

Finding Assignments

```
(R 4280 8
  (!S                                   NIL
   (!R    0                        8    4280
    (!R   16                       8    (LOGAND x y)
     (!R  (+ 40 (* 8 i))           8    111
      (!R 4280                     8    999
       ST))))))
```

$=$

Finding Assignments

```
(R 4280 8  ; [4280,4287] vs [0,7]
   (!R    0                  8     4280
    (!R   16                 8     (LOGAND x y)
     (!R  (+ 40 (* 8 i))     8     111
      (!R 4280               8     999
       ST))))))
```

=

Finding Assignments

```
(R 4280 8  ; [4280,4287] vs [16,23]
    (!R   16                 8    (LOGAND x y)
     (!R  (+ 40 (* 8 i))     8    111
      (!R 4280               8    999
       ST))))))
```

$=$

Finding Assignments

```
(R 4280 8  ; [4280,4287] vs [40,167] w/ i < 16
     (!R  (+ 40 (* 8 i))    8    111
      (!R 4280              8    999
       ST))))))
```

=

Finding Assignments

```
(R 4280 8  ; same!
      (!R 4280             8    999
       ST))))))
```

$=$

999

Preliminary Performance Results

A: guard verification

B: well-formedness

C: honsing

D: memoization

–	988 secs
A	955 secs
A+B	618 secs
A+B+C	494 secs
A+B+C+D	375 secs

Future Work

- provide a metafunction to prove state equality
- engineer ACL2 to cope better with large definitions

More Generally

This project illustrates a very common industrial application of ACL2: as a programming language suitable for writing verified programs.

By mixing verified metafunctions with the rest of ACL2, one can build a powerful domain-specific prover.