Extended Abstract: Stobj-tables

Matt Kaufmann, Rob Sumners, Sol Swords

ACL2 Workshop 2022

Single-Threaded Objects (`stobjs`)

- Added to allow ACL2 functions to operate on efficient structures..
  - ..fast access and destructive updates to arrays and hash-tables
- But ACL2 logic is defined on constructive lists..
  - ..so ACL2 has to put strict rules in place in the use of stobjs to keep fast execution consistent with logical meaning
- Example use: Quicksort (from [Ray, Sumners, 2002]):

```
(defstobj qst
  (objs :type (array ...) ...))
```

- Use of arrays in stobjs in ACL2 allows implementation of in-place Quicksort which requires fast array access and destructive updates in order to be efficient.

Single Threading?

- Simple: ACL2 requires a stobj to be accessed and bound to a variable of the same name..

```
(defun quicksort (lo hi QST)
  (declare (xargs :stobjs (QST) ...))
  (if (n<= hi lo) QST
    (mv-let (index QST) (split-qs lo hi (objsi lo QST) QST)
      (if (n<= index lo) (quicksort (1+ lo) hi QST)
        (let ((QST (quicksort index hi QST)))
          (quicksort lo (1- index) QST))))))
```

- Every update of a stobj overwrites the previous value ensuring destructive execution consistent with logic definition.

ACL2 Function Signatures

- In order to enforce stobj rules, ACL2 records function signatures that define how a function accesses and updates stobjs:

```
(objsi * qst) => *
(upadte-objsi * * qst) => qst

(split-qs * * * qst) => (mv * qst)
(quicksort * * qst) => qst
```

- Numerous extensions and improvements have been made to stobjs through the years..
  - local stobjs, nested stobjs, abstract stobjs, congruent stobjs, ...
  - Stobjs have become a fundamental part of writing efficient ACL2 functions..
  - ..But, the strict rules enforced by ACL2 can be a burden.

Issues with Stobjs and Program Refinement

- ACL2 supports a form of program refinement using attachments to constrained functions.
- As an example, consider the following process scheduler:

```
(encapsulate
  (((pick st)  => *) ((ready * st) => *) ((exec * st) => st)
   ((proc-ids) => *) ((rank * st)  => *))
  ...)
...
(defun run (st)
  (declare (xargs :stobjs (st) ...))
  (let ((p (pick st)))
    (if (ready p st) (let ((st (exec p st))) (run st))
      (report-completion-or-error-and-return p st))))
```

- Can change scheduler, add and remove processes through attachments and use the same run function..
- ..but ACL2 requires the stobj `st` to be fully defined and fixed.

Stobj-tables.. to the rescue!

- A stobj-table is a new type of field in a stobj which:
  - logically, is an alist that associates the name of a stobj with an instance of the stobj.
  - for execution, is a hashtable which represents mapping of stobj names to stobj instances.

```
(defstobj stobj-table (tbl :type (stobj-table)))
```

- Using nested stobj access, a stobj-table can store any stobj (even those yet to be defined).
- Stobj-tables have many potential uses: recursive stobjs, lazy creation of stobjs, wrapping any number of stobj parameters into a single structure, as well as supporting use of stobjs in program refinement.

Stobj-tables.. accessing via `stobj-let`

- Stobj-tables are accessed using `stobj-let` to pull a child stobj out of a stobj-table field in the parent:

```
(defstobj parent (tbl :type (stobj-table)))

(defstobj child fld)

(defun modify-child (parent)
  (declare (xargs :stobjs (parent)))
  (stobj-let ((child (tbl-get 'child
                              parent
                              (create-child))))
             (child)
             (update-fld (not (fld child)) child)
             parent))
```

Returning to the scheduler..

```
(defun run (st)
  (declare (xargs :stobjs (st) ...))
  (let ((p (pick st)))
    (if (ready p st) (let ((st (exec p st))) (run st))
      (report-completion-or-error-and-return p st))))
```

- Add a stobj-table field to the stobj `st`:
  - Scheduler and processes can be defined independently – each with their own child stobj to use for computation.
  - These child stobjs are stored in a stobj-table in the stobj `st`.
  - `books/demos/stobj-table-examples.lisp`
  - `books/workshops/2022/README`

Running the scheduler..

- The process scheduler is fully modular:
  - We can now add new processes (defined on new stobjs) using a macro `(add-pi i)` and execute the processes via a scheduler in our top-level run function..

```
ACL2 !>(include-book "examples")
ACL2 !>(add-pi 1)
ACL2 !>(add-pi 2)
ACL2 !>(reset-run st)
ACL2 !>(run st)
(COMPLETE <st>)
ACL2 !>(add-pi 3)
ACL2 !>(reset-run st)
ACL2 !>(run st)
(("process trying to release a lock but there is no lock!" P3 LOCK-P2)
 <st>)
```

Acknowledgements and Questions?

- We thank J Moore for feedback on this work and the paper.
- We also want to acknowledge that the work on stobj-tables was supported by ForrestHunt, Inc. and Centaur Technology, with continuing support from Intel.

Thank you!, and Questions?