Using SNARK as a Propositional Knowledge-Based System for CarPool World A Sample Project Report for CSE 4/563 Knowledge Representation

Stuart C. Shapiro
Department of Computer Science and Engineering
201 Bell Hall
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000
shapiro@cse.buffalo.edu

December 19, 2007

1 Introduction

This paper is a report on the use of SNARK (SRI's New Automated Reasoning Kit) [1, 2] and the `query` function of the Ask Interface for SNARK [4] as a propositional knowledge-based system for CarPool World [3]. SNARK is a resolution refutation theorem prover that may be given a knowledge base (KB) as a set of assertions, and then may be asked to prove that a well-formed proposition (wfp) is logically implied by that KB. SNARK either produces a refutation proof by deriving the empty clause, or reports that the empty clause cannot be derived. The `query` function of the Ask Interface assumes that the KB has been loaded, and then is given a comment in the form of a string and a query in the form of a wfp. The `query` function prints the comment, passes the wfp to the `ask` function, and prints the answer that `ask` returns. If SNARK can show that the KB logically implies the query, `ask` returns `True`; if SNARK can show that the KB logically implies the negation of the query, `ask` returns `False`; if neither, `ask` returns `Unknown`.

In Section 2 of this paper, the CarPool World domain is described. In Section 3, the syntax and semantics of the formalization is presented, along with the formalization of the domain rules and the specific data and questions of the demonstration run. Section 5 contains a transcript of the demonstration run, while the Appendix contains a listing of the file that is used for the demonstration run.

2 The Domain

The CarPool World domain is about two people, Tom and Betty, who drive to work together. It is described in [3] as:

> "Tom and Betty carpool to work.
> On any day, either Tom drives Betty or Betty drives Tom.
> In the former case, Tom is the driver and Betty is the passenger.
> In the latter case, Betty is the driver and Tom is the passenger." [3, p. 18]

Each day in CarPool World can be characterized completely by which of the following propositions are true, and which are false.

Betty drives Tom to work.
Tom drives Betty to work.
Betty is the driver of the car.
Tom is the driver of the car.
Betty is the passenger in the car.
Tom is the passenger in the car.

3 The Formalization

3.1 Syntax and Semantics of Atomic Propositions

CarPool World was formalized using the following atomic propositions with the given intensional semantics.

$[BettyDrivesTom]$	= Betty drives Tom to work.
$[BettyIsDriver]$	= Betty is the driver of the car.
$[BettyIsPassenger]$	= Betty is the passenger in the car.
$[TomDrivesBetty]$	= Tom drives Betty to work.
$[TomIsDriver]$	= Tom is the driver of the car.
$[TomIsPassenger]$	= Tom is the passenger in the car.

3.2 Domain Rules

The domain rules of CarPool World [3, p. 39], and their formalization in the SNARK syntax, using the atomic propositions shown in §3.1, are shown below.

1. Betty is the driver if and only if Betty is not the passenger.
   `(iff BettyIsDriver (not BettyIsPassenger)))`
2. Tom is the driver if and only if Tom is not the passenger.
   `(iff TomIsDriver (not TomIsPassenger))`
3. If Betty drives Tom, then Betty is the driver and Tom is the passenger.
   `(implies BettyDrivesTom (and BettyIsDriver TomIsPassenger))`
4. If Tom drives Betty, then Tom is the driver and Betty is the passenger.
   `(implies TomDrivesBetty (and TomIsDriver BettyIsPassenger))`
5. Tom drives Betty or Betty drives Tom.
   `(or TomDrivesBetty BettyDrivesTom)`

4 Example Questions

To test the adequacy of the formalization of CarPool World, the following questions were posed to SNARK. In the analysis of these questions, numbers refer to the domain rules listed in §3.2.

1. Is *"Tom is the driver or Betty is the driver"* implied by the domain rules?

   By 5, Tom drives Betty or Betty drives Tom.
   By 4, If Tom drives Betty, then Tom is the driver.
   By 3, If Betty drives Tom, then Betty is the driver.
   Therefore, Tom is the driver or Betty is the driver.
   So the answer should be "True".

2. Is `''Betty is the driver implies that Betty is not the passenger''` implied by the domain rules?

   By 1, Betty is the driver implies that Betty is not the passenger.
   So the answer should be "True".

3. Is `''Tom drives Betty and Tom is the passenger''` implied by the domain rules?

   By 4, If Tom drives Betty, then Tom is the driver.
   By 2, If Tom is the driver, then Tom is not the passenger.
   So either `''Tom drives Betty''` is False, or `''Tom is the passenger''` is False.
   So the answer should be "False".

4. Is `''Betty drives Tom''` implied by the domain rules?

   The domain rules do not imply either specific situation that `''Betty drives Tom''` or that `''Tom drives Betty''`.
   So the answer should be "Unknown".

Notice that these four questions include one wfp that is logically implied by the domain rules, one that is inconsistent with the domain rules, and one that is contingent in the situations specified by the domain rules. Moreover, every domain rule was used to answer at least one question. So these questions are an adequate test of the domain rules.

5 Demonstration Run

The following shows SNARK and the `query` function [4] being used to interact with a knowledge-based system that contains the CarPool World domain rules. The answers produced by SNARK agree with the answers derived in §4, showing the adequacy of the formalization shown in §3. The file that produced this demonstration is listed in the Appendix. This run has been edited only by:

- deleting some `snark` load messages;
- adjusting line breaks to make it easier to read;
- adding some blank lines, and deleting some others to make it easier to read.

```
cl-user(2): :ld /projects/shapiro/CSE563/ask
; Fast loading /projects/shapiro/CSE563/ask.fasl
;   Loading /projects/shapiro/CSE563/snark.cl

To start using SNARK, change the package to snark-user.
Then use (initialize), (assert <wff>), and (prove <wff>).

cl-user(3): :ld /projects/shapiro/CSE563/Examples/SNARK/CarPoolWorld
; Loading /projects/shapiro/CSE563/Examples/SNARK/CarPoolWorld.cl
; Running SNARK from
; /projects/shapiro/CSE563/snark-20070805r031/snark-system.lisp
; in International Allegro CL Enterprise Edition 8.1 [Linux (x86)]
; (Oct 16, 2007 16:39) on #xcd801823 at 2007-12-18T16:46:42

Ask if ''Tom is the driver or Betty is the driver''
    is implied by the KB.  (It is.)
  (ask '(or TomIsTheDriver BettyIsTheDriver)) = True
```

```
Ask if ''Betty is the driver
        implies that Betty is not the passenger''
    is implied.  (It is.)
  (ask '(implies BettyIsTheDriver (not BettyIsThePassenger))) = True

Ask if ''Tom drives Betty and Tom is the passenger'' is implied.
    (It's false according to the domain rules.)
  (ask '(and TomDrivesBetty TomIsThePassenger)) = False

Ask if ''Betty drives Tom'' is implied by the KB.
    (Neither it nor its negation is implied.)
  (ask 'BettyDrivesTom) = Unknown
```

References

[1] Mark E. Stickel, *SNARK - SRI's New Automated Reasoning Kit*, `http://www.ai.sri.com/~stickel/snark.html` (accessed December 18, 2007).

[2] Mark E. Stickel, Richard J. Waldinger, & Vinay K. Chaudhri, *A Guide to SNARK*, `http:www.ai.sri.com/snark/tutorial/tutorial.html` (accessed December 18, 2007).

[3] Stuart C. Shapiro, *Knowledge Representation and Reasoning: Logics for Artificial Intelligence*, Chapter 2, 2004, `http://www.cse.buffalo.edu/~shapiro/Courses/CSE563/Slides/chap2.pdf` (accessed December 18, 2007).

[4] Stuart C. Shapiro, *Ask Interface for SNARK*, `/projects/shapiro/CSE563/ask.cl`

Appendix

The demonstration file used for this paper is `/projects/shapiro/CSE563/Examples/SNARK/CarPoolWorld.cl`, the contents of which is shown below.

```
;;; Domain Rules for Propositional CarPool World
;;; for Loading into SNARK
;;; Stuart C. Shapiro
;;; February 2, 2004
;;; Modified for new query front end, defined in ask.cl 2/22/06

;;; Atomic Propositions
;;; ===================
;;; [BettyDrivesTom] = Betty drives Tom.
;;; [BettyIsTheDriver] = Betty is the driver.
;;; [BettyIsThePassenger] = Betty is the passenger.
;;; [TomDrivesBetty] = Tom drives Betty.
;;; [TomIsTheDriver] = Tom is the driver.
;;; [TomIsThePassenger] = Tom is the passenger.

(unless (find-package :snark-user)
  (load "/projects/shapiro/CSE563/ask"))

(in-package :snark-user)

(initialize)

;;; Betty is the driver if and only if Betty is not the passenger.
(assert '(iff BettyIsTheDriver
          (not BettyIsThePassenger)))

;;; Tom is the driver if and only if Tom is not the passenger.
(assert '(iff TomIsTheDriver
          (not TomIsThePassenger)))

;;; If Betty drives Tom
;;;    then Betty is the driver and Tom is the passenger.
(assert '(implies BettyDrivesTom
          (and BettyIsTheDriver TomIsThePassenger)))

;;; If Tom drives Betty
;;;    then Tom is the driver and Betty is the passenger.
(assert '(implies TomDrivesBetty
          (and TomIsTheDriver BettyIsThePassenger)))

;;; Tom drives Betty or Betty drives Tom.
(assert '(or TomDrivesBetty BettyDrivesTom))
```

```
;;; Sample Questions
(query
 "Ask if ‘‘Tom is the driver or Betty is the driver’’
    is implied by the KB.  (It is.)"
 ’(or TomIsTheDriver BettyIsTheDriver))

(query
 "Ask if ‘‘Betty is the driver
        implies that Betty is not the passenger’’
    is implied.  (It is.)"
 ’(implies BettyIsTheDriver (not BettyIsThePassenger)))

(query
 "Ask if ‘‘Tom drives Betty and Tom is the passenger’’ is implied.
    (It’s false according to the domain rules.)"
 ’(and TomDrivesBetty TomIsThePassenger))

(query
 "Ask if ‘‘Betty drives Tom’’ is implied by the KB.
    (Neither it nor its negation is implied.)"
 ’BettyDrivesTom)
```