Teaching Strategy - Constraint Propagator Interface

Intelligent Book Project

MIT Project for Mathematics and Computation

Cambridge Computer Laboratory

Summer 2003

This document contains all the APIs that talk to the teaching strategy program. Currently, they are implemented to print text. Ultimately, they should be able to do two things:

1. html manipulation (radio buttons, text etc)
2. diagram manipulation (jade, light up R_C)

```
;;When nothing can be further deduced
(define (reply-to-student-set-variable-quiescent ckt)
  (write-line "Nothing can be further deduced.  You need to set another unknown."))
;;No contradiction
(define (reply-to-student-no-contradiction ckt)
  (write-line "Accepted"))
;;State contradiction
(define (reply-to-student-state-contradiction ckt)
  (display "Your setting caused a contradiction:
n"))
```

```
;;Proof contradiction

(define (reply-to-student-with-proof contradiction)

  (write-line `(,(explain contradiction))))

;;Supports for contradiction

(define (reply-to-student-with-support support-lst)

  (write-line `(The contradiction is supported by these assumptions))

  (write-line `(,support-lst)))

;;These are the supports set by the student

(define (reply-to-student-with-support-retract-choice lst)

  (write-line `(Please retract one of these supports that you have set))

  (write-line `(,lst)))

;;Value set is correct

(define (reply-to-student-set-value-correct ckt path)

  (write-line `(value set is correct))

  (write-line `,(the-value ckt path)))

;;Value set is incorrect

(define (reply-to-student-set-value-incorrect ckt)

  (write-line `(value set does not match the value from the

      constraint  propagator, try again)))

;;Invalid retraction

(define (reply-to-student-invalid-retraction ckt)

  (write-line `(Cannot be retracted)))

;;All interesting variables have been set

(define (reply-to-student-completed-setting-variable ckt)

  (write-line `(All interesting variables have been set)))

;;Suggestion of a variable

(define (reply-to-student-with-variable connector)

  (write-line `(You can try to determine ,(get-path connector))))
```

```
;;All the variables that have been set and their corresponding values

(define (display-status lst)

  (if (null? lst)

      'done

      (begin

(pp `(,(get-path (car lst)) = ,(get-assignment-value (car lst))))

(display-status (cdr lst)))))

;;Listing all interesting variables

(define (reply-to-lst-all-vars ckt)

  (pp `(Variables are ,(map get-path interesting-var-lst))))

;;The value of a particular variable

(define (reply-to-student-with value ckt path)

  (the-value ckt path))

;;Variable that doesn't have a value yet (hasn't been set)

(define (reply-to-student-without-value ckt path)

  (write-line `(,path has not been set yet)))
```