Exercises, Responses, and Assessments

# Overview

exercise = problem + solution

problem = list of tasks + other supporting data

solution = list of task solutions + global constraints

response = (student’s) list of task responses

assessment = overall assessment + list of task assessments

|  |
| --- |
| ExerciseAnswerAssessment.png |

task solution = partially ordered set of steps + constraints

step = option set or atom

option set = set of options

option = (partially) ordered set of steps

atom = action, family, collapsible, or idiom

|  |
| --- |
| TaskSolution.png |

Note: The above diagram does not imply an inheritance hierarchy. Steps are OptionSet or Atom Steps through composition, not inheritance. Atoms reference actions, idioms, etc.

A task solution can be viewed as a directed acyclic graph that implicitly corresponds to a set of task solution instances.

|  |
| --- |
| Step.png |

response = list of task responses

task response = list of atoms (= student’s demonstration to accomplish task)

assessment = overall assessment + global constraint violations + list of task assessments

task assessment = alignment (to a particular task solution instance) + constraint violations

|  |
| --- |
| Alignment.png |

*benign atoms* = set of atoms considered “benign” by default (i.e., irrelevant to the quality of a solution); all other atoms are considered “potent” by default (i.e., relevant to the quality of a solution)

# Model Definitions

**\* Bold text denotes fields included in initial prototype**

# Exercise

* **id (*UUID*):** unique identifier
* **name (*string*): Descriptive name of exercise.** May not be displayed in all "uses" such as during a test
* **description (*string*):** Describes the exercise in the abstract (should be targeted towards the students and instructors, regardless of "use")
* **html (string):** HTML representation of the exercise for display purposes
* **problem (*Problem*):** ordered set of tasks
* **solution (*Solution*):** the gold standard (with references to tasks), including any information affecting assessment
* purpose (*string? Classification?*): The goal of this exercise (e.g., Teach mirror vs. clone)
* keywords (*set*<*string*>): keywords to aid instructors and students in finding appropriate exercises
* author (*string*): Name of the instructor who authored the exercise (useful in the event of a mistake in the exercise)
* difficulty (*enum* [EASY|MEDIUM|HARD]): describes difficulty of exercise to aid instructors and students in finding appropriate exercises
* uses (*set*<*enum* [TEST|TUTORIAL|EXERCISE|SCRIPT]>)

## Problem

* **tasks (*list*<*Task*>):** The tasks that comprise the problem

## Task

* **id (*string*):** unique identifier relative to Problem
* **description (*string*): brief description of the task**
* **prompts (List<Prompt>):** And optional list of prompts that can be used to guide the student through this task
* timeWarningDelay (duration): the time to elapse before warning the student about taking too long

## Prompt

* **text (*string*):** the prompt text
* hintDisclosureDelay (duration): the time to elapse before providing the student with a hint
* hint (*string*): An optional hint the instructor may want to optionally disclose
* helpTopics(List<URL>): Relevant help topics for the current prompt

## Solution

* **taskSolutions (*list*<*TaskSolution*>):** the solution components for each Task in the problem
* **equalityConstraints (*set<ValueEqualityConstraint*>):**  Contains any global equality constraints; generally used to check support relationships across tasks
* maximumTime (duration): Maximum time that may elapse during a student’s response and still considered a successful response
* maximumAtoms (int): Maximum number of atoms that may appear in a student’s response before crossing the threshold from suboptimal to failure.
* minimumScore (float): The minimum score that may be considered a success or alternatively, the minimum score required to allow the student to proceed
* benignAtomOverrides (map<string, boolean>): Allows the global set of benign atoms to be overriden for this Solution. The key is the functor (string) of the atom. If the value is true then the functor should be considered benign. If it is false then the functor is “potent”.
* prohibitedAtomConstraints (set<ProhibitedAtomConstraint): Atoms that should be considered prohibited for this solution

## TaskSolution

* **task (*Task*):** the Task in the Problem for which this is a solution
* **option (Option):**  a possibly constrained partially ordered set of steps that embodies the solution to *task*
* maximumTime (duration): Maximum time that may elapse during a student’s response and still considered a successful response
* benignAtomOverrides (map<string, boolean>): Allows the global set of benign atoms to be modified locally for this TaskSolution. The key is the functor (string) of the atom. If the value is true then the functor should be considered benign. If it is false then the functor is “potent”. Specifications in this field trump specifications in Solution.benign\_atom\_overrides

## Option

* **steps (set<Step>):** (partially) ordered set of Steps that collectively contribute towards a solution instance
* **equalityConstraints (*set<EqualityConstraint*>):**  Contains any equality constraints across *steps*
* **orderingConstraints (*set<OrderingConstraint*>):** The ordering constraints that apply to *steps*
* **queryConstraints (*set<QueryConstraint*>):** query constraints that that involve values from one or more step in *steps*
* **valueConstraints (*set<ValueConstraint*>):** constraints on the values of parameters defined by steps in *steps*
* description (*string*): brief description of how the option accomplishes the goal
* prohibitedAtomConstraints(set<ProhibitedAtomConstraint>): Atoms of which an unexpected instance will constitute an error
* setConstraints (*set<ValueSetConstraint*>): Contains any value set constraints
* rangeConstraints(setValueRangeConstraint>): Contains any value range constraints

## Step

* **id (*string*):** unique identifier relative to Option
* **atom (Atom):** the Atom if this is an atom Step
* **option\_set (OptionSet):** the OptionSet if this is an OptionSet step
* Note: each step will comprise either an Atom or an OptionSet, not both.

## Atom (represents an action, family, collapsible or idiom)

* **functor (string):** The functor of the atom
* **parameters (*set*<*Parameter*>):** inputs required by the Option

## OptionSet

* **options (set<Option>):** a set of alternatives step sequences, each of which contributes towards a solution instance
* preferredOption (*Option*): the preferred option (e.g., to support tutorial playback)

## Parameter

* **id (*string*):** Identifier for the parameter. Must be unique within its scope
* **accessor (string):** The “accessor” for the parameter. For a top-level parameter this corresponds the name of the action argument the parameter refers to. For sub-parameters this will be either an element index (for lists), a well-known collection accessor (first, last, only), or a field name for structs.
* **subParams(set<Parameter>):** A list of “sub-parameters” of this parameter. In the case of collections this refers to specific elements within the collection. In the case of a struct it refers to specific fields within the struct.
* Note: If a constraint references a parameter that is bound within an OptionSet step, then every option within the OptionSet must bind the parameter id.

## OrderingConstraint

* **predecessor (*Step*):** The step that should precede the successor
* **successor (*Step*):** The step that should succeed the predecessor

## QueryConstraint

* **functor (*String*)**: the functor of the query action to call
* **arguments (*list*<*QueryArgument*>)**: the parameters and/or values involved in the query; may be just one parameter

## QueryArgument

* **parameter (String)**: The idof a Parameter whose value should be used in the evaluation of the QueryConstraint
* **value (String):**  a CTR-S encoded value to be used in the evaluation of the QueryConstraint
* Note: each QueryArgument will have either a parameter reference or a value, not both

## ValueConstraint

* **parameter (*String*):** The id of the parameter whose value is being constrained
* **negated (Boolean):** Whether or not the value constraint should be negated
* **value (*ATRTerm*)**: the value the parameter should have represented as an ATRTerm. Will be encoded in XML as CTR-S.

## ProhibitedAtomConstraint

* functor (string): the functor of the prohibited atom
* Note: A prohibited atom constraint may appear on an option that contains references to the prohibited atom. In this case, it is assumed that any extraneous (non-matching) instances of the atom constitute an error but the others do not.

## RepetitionConstraint

* min (*integer*): min #times the step must be performed
* max (*integer*): max #times the step can be performed
* Note: *RepetitionConstraint(0,\*)* = *optional*; *RepetitionConstraint(0,0) = Prohibited*

## ValueSetConstraint

* parameter (*Parameter*)
* values (*set*<ATRTerm>): the set of ATRTerms the parameter is constrained to. The terms will be encoded in XML as CTR-S.

## ValueRangeConstraint

* parameter (*Parameter*)
* minValue (*number*): the lowest value the (integer) parameter can have
* maxValue(*number*): the highest value the (integer) parameter can have
* Note: undefined (or ‑∞) *min\_value* covers *param < value*; undefined (or +∞) *max\_value* covers *param > value*

# Response

* **id (*UUID*):** unique identifier
* **exercise (*Exercise*):** the exercise being answered
* **taskResponses (*list*<*TaskResponse*>):** the student’s response for each task in the exercise
* student (Student): The identity of the student who made this response

## TaskResponse

* **demonstation (ATRDemonstration):** the trace of the student’s actions for the current task (already processed for/converted to *idioms* and *collapsibles*)
* **timeElapsed (duration):** how long each action took
* numActions (int): total # of actions taken by user for this task response. This field might be necessary because collapsible actions will have been collapsed in the ATRDemonstration

# Assessment

* **taskAssessments (list<TaskAssessment>):** The individual assessments for each task
* **globalEqualityIssues (*set<EqualityIssue*>):** Equality/identity issues that span tasks

## TaskAssessment

* **equalityIssues(set<EqualityIssue>):** Ordering issues with the student’s response to a task
* **orderingIssues(set<OrderingIssue>):** Ordering issues with the student’s response to a task
* **queryIssues(set<QueryIssue>):** Ordering issues with the student’s response to a task
* **valueIssues(set<ValueIssue>):** Ordering issues with the student’s response to a task

## ArgumentLocation

* **atomIndex (integer):** the 0-based index of an atom in a Response. Note that the index spans across TaskResponse demonstrations.
* **accessors (List<String>):** List of string “accessors” to navigate directly to the argument in question. See definition of Parameter for more information.

## EqualityIssue

* **location1(ArgumentLocation):** First argument location of a failed equality constraint
* **location2(ArgumentLocation):** Second argument location of a failed equality constraint
* **support(Boolean):** Whether or not the equality constraint represented a support

## OrderingIssue

* **Details still being worked out**

## QueryIssue

* **locations(list<ArgumentLocation>):** the locations of any arguments involved in the query Issue. Should be in the order the arguments were passed to the query constraint.
* **Constraint(QueryConstraint):** The constraint that led to identification of the Issue

## ValueIssue

* **location (ArgumentLocation):** The location of the value Issue
* **constraint (ValueConstraint):** the failed value constraint

## ExtraAtomsIssue extends LocalizedIssue

* startAtomIndex (integer): the index of the first atom in a sequence of extra atoms (length >= 1)
* endAtomIndex (integer): the index of the last atom in a sequence of extra atoms

## MissingAtomIssue extends LocalizedIssue

* displayIndex (integer): Index at which to display the missing atom (should correspond to a legal location where it “should” have appeared)
* functor(String): The functor of the missing atom