Interactive Student Error Detection and Correction
With the application on Lab Experiments

Magdy Aboul-Ela
Sadat Academy for Management Sciences
Computer and Information Systems Department
P.O. Box 2222, Kournich-ElNile, El-Maadi, Cairo, Egypt

ABSTRACT: This paper describes an interactive model for the student error detection and correction in the e-Learning environment, which is designed to provide feedback on the student action, observations, or explanations with the application on Lab experiments. In case of an erroneous response from a student, the program notifies the student with each mistake and its correction. The error detection is achieved by using the three major components: a specific knowledge structure to hold the representation of the meaning of the relevant objects, actions, and the relations with the facts and rules of the subject domain; a Knowledge Description Language (KDL) to support in expressing the knowledge by the knowledge engineer, or by the student to express his observations and explanations; and the Interactive Analysis Method (IAM) with the difference algorithm for error detection and correction.

KEYWORDS: Human Computer Interaction, e-Learning, Knowledge Representation, Student Model, Intelligent Tutoring System, Student Error Detection

1. Introduction

The essential requirement to achieve student model is focused on knowledge description, and autonomous error detection.

Expert model tutors incorporate production rules associated with error free and efficient task performance. These systems intervene with corrective feedback as soon as a student deviates from a solution path. While the intelligent novice tutor allows students to make errors, and provides guidance through the exercise of error detection and correction skills. The underlying cognitive model in such a tutor includes both rules associated with solution generation, and rules relating to error detection and correction [1].

This paper presents a proposed model of error detection and correction based on knowledge understanding, which propose a Knowledge Descriptive Language (KDL), and the Knowledge Structure, to represent the actions, objects, relations, facts, and rules of the subject domain, with an Interactive Analysis Method of student’s knowledge to detect the errors and make an instant verification, for the case study physics experiments. The model provides a feedback on the basis of detecting the disconnection, or gap, between the consequences states of the problem, due to disorders, missing, or wrong use of the actions; and if this gap can be filled implicitly. Instead of using associated rules for solution generation or error detection.

2. Knowledge Description Language (KDL)

A modified version of the knowledge structure [2], [3] represents the variance types of knowledge. It is based mainly on the representation of any entity or action by its attributes and its relation to other entities. In case of the action, a script or the scenario of event or action is represented.

The knowledge representation language, such as method-description language [4], Domain description Language [5], action description language [6], allows users to specify the basic components of a domain, to associate a description to those components and to specify the constraints and mutual relationships among components. Matching the problem and knowledge base to a method in the reuse library, needs an understanding of these methods.

The proposed knowledge description language KDL, which incorporates objects, actions, attributes, and attributes value, includes conditionals (when/if-;then), routines (repeat),
and user defined functions for specific domain tasks. A complete and implicit translation of domain descriptions written in KDL into the knowledge structure in the form of objects, attributes, relations, Facts, rules, actions, and plans, to represent the meaning of theses knowledge elements. KDL is used to build the knowledge base and by the student to express his idea or understanding, without need to use natural language in any level.

The following grammar represents the knowledge description language:

```
<Statement> :: <rule> | <fact> | <object description> | <attribute description>

<rule> :: <condition article> <result>

<fact> :: <object-pattern> {<relation> | <action>} [object-pattern] | <expression>

<result> :: <fact> | {<reference> | <object-pattern>} {become | <object-pattern>} [and | result]

<condition> :: <fact> [and | <condition>]

<object-Pattern> :: [attribute] of | <quantity> | <number>[<symbol>]

<attribute-pattern> :: <attribute-value> [attribute]

<adjective-attribute> :: <attribute> adjective-attribute | <attribute>

<part-attribute> :: <object> st: <object> part-relation LAST.

<attribute description> :: <attribute> : <attribute values List>

<attribute values List> :: <attribute> : <attribute Value>[<attribute values List>]

<object description> :: <object> : <attributes list>

<object List> :: <object> [<object List>]

<action description> :: <action> : [<attribute-values List>][script: <roles>:

<object List>] [Condition: <condition>], Result:<result>[Result: <plan> [repeat] [<action description list] <object List> ] ]

'action description list> :: <action description>[<action description list>]

<condition article> :: <condition article> When | if

<conjunction article> :: ; | then

<reference> :: <first-object-reference> | <group-object-reference>

<first-object-reference> :: single-reference-pronoun // e.g. it

<group-object-reference> :: plural-reference-pronoun // e.g. they
```

Where:

:: means defined by

<...> for non-terminal,

[... ] for option,

{...} must be included,

| for ‘or’.

St: means subject to.

// for comment.

And the keywords: LAST means the last non-terminal at the statement, MEANING IS refers to the meaning of the preceding word in a semantic lexicon or the knowledge structure.

3. Interactive Analysis Method (IAM)

The objective of this method is to find the isolated references for objects and the disconnection between actions by searching in the knowledge structure for the experiments, which is created through Interactive User Interface Scenario.

Interactive User Interface Scenario:

1. Select action/object
2. Create new knowledge element to represent the state of action/object to be part of problem state.
3. Get the relevant attributes list
4. Fill attributes value in knowledge structure
5. Verify the knowledge:
   a. Types Mismatch
   b. Verify facts/rules
   c. Apply rules and add the new facts.
   d. Update knowledge structure
6. Repeat steps from 1 to 5 until experiment is done from the student view.

Interactive Analysis Method (IAM):

Evaluation scenario of the experiment steps:

1. Trace the steps
2. Find the difference between the consequence steps
3. Evaluate the difference, if it is acceptable preprocessing or not regarding:
   a. Missing action
   b. Missing object
c. Missing attribute value

d. Type mismatch

e. Incomplete steps or disconnection.

4. If the difference is not acceptable analyze the error and notify the user.

The Difference Algorithm:

The problem state, which reflects the states of all objects and the current facts may be changed after each action/rules/facts, and can be traced to check the validity of changes to the applied action/rules/facts:

\[
\text{State}_0(O_0, \ldots, O_n, F_0, \ldots, F_n) \rightarrow (\text{action}) \rightarrow \text{State}_1(O_0, \ldots, O_n, F_0, \ldots, F_n) \rightarrow (\text{Rules/Facts}) \rightarrow \text{State}_2(O_0, \ldots, O_n, F_0, \ldots, F_n) \rightarrow (\ldots) \rightarrow \text{State}_n(O_0, \ldots, O_n, F_0, \ldots, F_n)
\]

Algorithm:

1. Repeat for all action {
   a. Get next action as the current action.
   b. Apply current action on current state
   c. Apply rules on the current states using forward chaining and add the new facts to the current state.
   d. Get the next action in sequence from student actions list to be the current action; if it is null, exit; otherwise continue.
   e. Check the condition of the current action to the current state, if it is matched, continue, otherwise, refuse this action with the notification of the missing facts. }

2. Repeat for all facts/rules in student explanation list {
   a. get the next fact/rule as the current fact/rule.
   b. if it is a fact, check its existence in the problem state.
   c. if is a rule, check the existence of the facts of the rule condition in the facts at the student explanation list; if it is matched, fire it from the explanation list and check it in the problem state; if it matched, continue, otherwise, refuse this fact.
   d. in case of all facts at condition are matched, check the result fact; if is matched with the current problem state }

The matching between two facts is based on associate the attribute lists to the objects in the fact to assure that any adjectives of the objects in the fact are considered. For example: small uncharged material, and the plastic ruler attributes include charged: no, weight: small, and after evaluating the values small and uncharged to know the attribute names to get the corresponding one from the attribute list of the object.

4. Physics Experiment Example:

Consider the following piece of knowledge for a physics lab experiment:

**Electric charge:**

*When rubbing two different insulating materials against each other they become electrically charged.* A substance that gains electrons becomes negatively charged, while a substance that loses electrons becomes positively charged. Positive and positive charges or negative and negative charges repel, while positive and negative charges attract. Charged objects are able to attract small uncharged objects. When the materials are rubbed against each other:

- Electrons move from one material to the other.
- The material that loses electrons becomes positively charged.
- The material that gains electrons becomes negatively charged.
- Both materials gain an equal amount of charge, but the charges are opposite.
- The electrically charged objects may attract or repel each other, and that they can attract small uncharged object.

The following is the representation of the relevant knowledge in the form of Objects, Actions, Attributes, Relations, Facts, and the Rules according to the above Knowledge Structure:

**Objects:**

*The knowledge representation of the objects according to the knowledge structure:*

**Object Name:** [attributes List]

Each element of the attribute list point to the attribute-value pair inherited from attribute class.

*For example:*

**Material/substance:** [Conductivity, Naturally, Charging, Size, Weight, Shape, Quantity, Boundary, Position]

**Charge:** [Charge Type]

**Point:** [Position, Belongs-to]

**Proton:** [has: positive charge]

**Neutron:** [has: no charge]

**Electron:** [has: negative charge]

**Paper:** [isa: material]

**Attribute:**

*Attribute Name:** [attribute values list]

*For example:*

**Conductivity:** [Domain: Electricity; Range: Poor | Medium | High | Not]
Naturally: [Range: Poor | Medium | High | Not]
Charging: [Range: Poor | Medium | High | Not]
Size: [Range: small | medium | large]
Shape: [Range: regular | irregular]
Quantity: [Range: counts | amount-of | number-of]
Charge Type: [Range: positive | negative]
Position: [Range: in | on | between | below | at]
Agent: [Range: human | material]
Object: [Range: human | material]
Object-1: [Range: human | material]
Object-2: [Range: human | material]

Action:
- Action Name: [attributes-value List], script: roles: [objects List],
  Condition: expression, Result: expression, Plan: actions List.

For Example:
- Rub: agent: human, object-1: material, object-2: material, condition: object-1 touch object-2,
  script: repeat { (object-1 touch object-2), move object-1}
- Charge: agent: human, agent: beneficiary: material, object: charge, mean: rubbing | electricity
- Conduct: agent: material, object: electricity or heat, Beneficiary: material, mean: material or touch
  condition: (agent touch mean, mean touch beneficiary) or (agent touch beneficiary).
- Repel: agent: material, object: material,
  Condition: object close-to agent, Result: object move-away agent
- Attract: agent: material, object: material,
  Condition: object far-of agent, Result: (object move-to agent) or (object touch agent)
  Result: Difference (object-1. Position, object-2. Position) = 0, Plan: repeat {MoveTo: object-1, object-2 }
- Move: agent: student, object: material, Plan: change(object.position, Newposition),
  FreePlace(Newposition), ChangeAttribute(object.position, Newposition).
- MoveTo: agent: student, object-1: material, object-2: material, Condition: object-1 far-of object-2,
  Result: object-1 close-to object-2, Plan: repeat { move object-1).
- MoveAway: agent: student, object-1: material, object-2: material, Condition: object-1 close-to object-2,
  Result: object-1 far-of object-2, Plan: repeat { move object-1)
- Change (object.attribute, [object.attribute – delta]).
- ChangeAttribute (object.attribute, NewattributeValue).

Facts:
- <object pattern>{ <relation> | <action>}

For Example:
- insulating Material isa poor conductor.
- conductor conduct electricity or heat.
- insulator isa electrically charged.
- positive charge repel negative charge.
- positive charge attract positive charge.
- negatively charged material attract small uncharged material.
- pieces of paper isa uncharged material.
- pieces of paper isa insulating material.

Rules:
For Example:
- When insulating material rubs insulating material; they become electrically charged.
- When substance loses electrons; it becomes positively charged.
- When insulating material A rubs insulating material B; A loses electrons and B gains electrons.
- When material loses electrons it becomes positively charged.
- When material gains electrons it becomes negatively charged.

Relation:
For Example:
- Far-of: object-1, object-2,
  Difference (object-1. Position, object-2. Position) > 0
- Close-to: object-1, object-2,
  Difference (object-1. Position, object-2. Position) < delta
- Part-of: object-1, object-2,
  Find (object-1, GetAttributeList(object-2))
- Composed-of: object-1, object-2,
  Find (object-2, GetAttributeList(object-1))
- Difference: object1.attribute, object2.attribute,
  Calc (object1.attribute-object2.attribute)
- Between: object, object-1, object-2,
  Intersect (object.position, Path(object-1. Position, object-2. Position))
- FreePlace: object
  NoObjectExist(object.position)

Where the functions difference (), find (), calc (), and intersect () are user defined functions depend on the relevant domains. The user can
define any function and add it to the system
library.

**Experiment Steps:**

The student is asked to perform the experiment by selecting the relevant objects, actions, and the sequences steps with explanation of the observation. The system analyzes all these steps with the explanation; and tries to find if there is any error, and notifies the student with correction if this is possible.

**Selected Objects:**
The user selects the objects and the system creates the attributes list as defined with object definition. Consider the user selects the following objects:

- Material: Plastic Ruler
- Material: paper
- Material: wool cloth

**Selected Actions:**
Also, the user select the actions in sequences, and the system will create the attribute lists as defined in the action definition. Consider the user selects the following actions in sequences:

- moveTo: agent: student, object-1: plastic ruler, object-2: wool cloth
- touch: agent: student, object-1: plastic ruler, object-2: wool cloth,
- rub: agent: human, object-1: plastic ruler, object-2: wool cloth,
- moveAway: agent: student, object-1: plastic ruler, object-2: wool cloth
- moveTo: agent: student, object-1: plastic ruler, object-2: pieces of paper

**Interactive Analysis Method for Experiment Steps:**
The partial relevant knowledge structure for this experiment is as following, where the system looks for the holes and mismatch in the knowledge elements, and the disconnection or the disorder between events:

**Problem States:
Stat 0:**
The objects states:

- Plastic Ruler:
  - Isa material, isa: solid, Conductivity: Not, Naturally: Not, Charging: Not,
  - quantity: 1, size: small, weight: small, position: four control points

- Paper:

- Wool cloth:
  - Isa material, isa: solid, isa: textile,
  - Conductivity: Not, Naturally: Not, Charging: Not,
  - quantity: one pieces, size: small, weight: small, position:
  - four control points

**1st action**
- Concept: Name: touch, Domain: Electricity, Attribute List
- Event/Action: touch, Isa touch, Script: touch
- Script: Script-Name: touch
- Roles: [plastic ruler, wool cloth]
- People: [student]
- Condition: Difference (object-1. Position, object-2. Position) > 0
- Result: Difference (object-1. Position, object-2. Position) = 0
- Plan: repeat {moveTo: object-1, object-2 }

- Where moveTo action is represented as follows:
- Event/Action: moveTo, Isa moving, Script: moveTo
- Script: Script-Name: moveTo, Roles: [plastic ruler, wool cloth]
- People: [student]
- Condition: object-1 far-of object-2, No object between object-1, object-2
- Result: object-1 close-to object-2

**State 1:**
New fact is added and changing in the value of the attribute position of the plastic ruler object:
- Plastic ruler close-to wool clothes.
- Plastic ruler touch wool clothes.

**2nd action**
- Concept: Name: rub, Domain: Electricity, Attribute List
- Event/Action: rub, Isa rub, Script: rub
- Script: Script-Name: rub
- Roles: [plastic ruler, wool cloth], People: [student]
- condition: object-1 touch object-2, plan: repeat { object-1 touch object-2, move object-1}

**State 2:**
New fact is added after the 2nd action:
- Plastic ruler rub wool cloth

Plastic ruler and wool cloth super classes can replace the objects and new fact is added:
- Insulating material rub insulating material.

Applying forward chaining on the rules, new facts are added, with the change in the attribute values of the objects:
- Plastic ruler isa electrically charged.
- Insulating material isa electrically charged.

**3rd action**
- Event/Action: moveAway, Isa moving, Script: moveAway
Script: Script-Name: moveAway, Roles: [plastic ruler, wool cloth]
People: [student]
Condition: Object-1 close-to object-2
Result: object-1 far-of object-2
Plan: repeat { move object-1)

State 3:
The new fact is: plastic ruler far-of wool cloth.

4th action
Event/Action: moveTo, Isa moving, Script: moveTo
Script: Script-Name: moveTo
Roles: [plastic ruler, pieces of paper]
People: [student]
Condition: object-1 far-of object-2,
No object between object-1, object-2
Result: object-1 close-to object-2

State 4:
The new fact: Plastic ruler close-to pieces of paper
Pieces of paper isa uncharged material
From the available facts:
Insulating material attract small uncharged material.
Plastic ruler attract pieces of paper

Observation:
The student describes the observation, using selected objects and can select actions:
Plastic ruler attract pieces of paper.

This observation matched with the last state.

The relevant knowledge structure:
Event/Action: attract, Isa attract, Script: attract
Script: Script-Name: attract
Roles: [plastic ruler, pieces of paper], People: [student]
Condition: object far-of agent,
Result: object moveTo agent or object touch agent

Explanation:
The student can give his explanation using the available objects list and actions list, and the pattern structure of the sentence:
insulator material becomes charged when insulating material rub insulating material.
plastic ruler isa insulator material.
pieces of paper isa insulator material.
pieces of paper isa uncharged material.
charged material attract uncharged material.

When the user facts are matched with the last state of the problem, this means the explanation is right, otherwise the student does not aware with the reasoning of the observation.

5. Conclusion
The proposed model for Interactive human error detection-correction combines the internal representation of the knowledge and the problem states with a knowledge description language, and the analytical method, which analyze the problem states with the difference algorithm to find the available mistakes either in the experiment steps, or the explanation. The model can be incorporated in a collaborative learning environment, where many users can share the experiments and see the mistakes, without intervene of the teacher. The model can be enhanced by considering the similarity degree in matching between the attribute values, facts, and actions.

REFERENCES: -