A Graphics-User Interface In Support of a Cognitive Inference Architecture

Isaí Michel Lombera[1], Jayeshkumar Patel[2], Stuart Rubin[3], Shu-Ching Chen[4] and Gordon Lee[1]

[1]Dept. of Electrical & Computer Engineering
San Diego State University
5500 Campanile Drive
San Diego, CA 92182

[2]Dept. of Computer Science
San Diego State University
5500 Campanile Drive
San Diego, CA 92182

[3]SPAWAR - Systems Center
53560 Hull Street
San Diego, CA 92152
email:
stuart.rubin@navy.mil

[4]Dept. of Computing and Information Sciences
Florida International University
Miami, FL 33199
email: chens@cs.fiu.edu

Abstract **-** The objective of this paper is to present a graphical-user-interface (GUI) in support of a decision support system (KASER) for machine understanding. In order to provide information between the user and the KASER during the learning process, the approach is to combine information science and cognitive science in the form of several virtual and physical multimedia screens (e.g., a whiteboard running a pen-based OS ensemble, a menu-driven touch-screen, or a multimedia output screen). The touch-screen will also facilitate navigation between virtual screens. An application to homeland security is provided as an example; however the approach has vast applicability to many problems in which a graphical form of learning is required.

The integration between human and machine is to be seamless which allows the user to pose questions and retrieve answers through the multimedia system, thus accelerating the learning process.

1. INTRODUCTION

The KASER is a knowledge amplifier (the acronym stands for *K*nowledge *A*mplification by *S*tructural *E*xpert *R*andomization) based on the principle of randomization. This principle refers to the use of fundamental knowledge in the capture and reduction of a larger, dependent space of knowledge (not excluding self-reference). In a KASER system, the user supplies declarative knowledge in the form of a semantic tree using single inheritance. Unlike conventional intelligent systems, however, KASERs are capable of accelerated learning in symmetric domains [1-2].

Conventional expert systems generate cost curves below the breakeven line. In conventional expert systems, cost increases with scale and the increase is never better than linear. In the case of KASER systems, the cost *decreases* with scale and is always better than linear, unless the domain is asymmetric (random). Perfectly (asymmetric) random domains are trivial constructs and are not encountered in the construction of practical applications [3].

Conversely, perfectly symmetric (non-random) domains are also trivial and are also not found in practice [3]. In other words, a perfectly random domain would have no embedded patterns (true random numbers), while a perfectly symmetric domain would be infinitely compressible (free of information content). Clearly, such constructs are strictly artificial. The more symmetric is the operational domain, the less the cost of knowledge acquisition.

As a synopsis of the KASER, a production rule is defined to be an ordered pair whose first member is a set of antecedent predicates and whose second member is an ordered list of consequent predicates. Predicates can be numbers or words [4-5]. The linking of the two members forms rules or courses of action.

KASER systems can be classified as Type I and Type II, depending on their characteristics. In a Type I KASER, words and phrases are entered through the pull-down menus. The user is not allowed to enter new words or phrases if an equivalent semantics already exists in the menu. In a Type II KASER, distinct syntax may be equated to yield the equivalent normalized semantics. The idea in a Type II KASER is to ameliorate the inconvenience of using a data entry menu with scale. In a Type II KASER, selection lists are replaced with semantic equations from which the list problem is automatically solved.

Thus a KASER system can amplify a knowledge base. It represents an advance in the design of intelligent systems because of its capability for symbolic learning and qualitative fuzziness. In a conventional expert system, the context may cover the candidate rule antecedent, in which case an agenda mechanism is used to decide which matched rule to fire (most-specific match, first to match, chance match.). The KASER system follows the same rule-firing principle – only the pattern-matching algorithm is necessarily more complex and embeds the conventional approach as its degenerate case.

In order to transmit and receive information back and forth between the user and the KASER system in a symbiotic manner, a novel graphics-user-interface has been designed. We note that the GUI plays an important role in supporting learning for the KASER through the user. In fact, this synergy accelerates learning through visualization.

2. THE GUI PHILOSOPHY

The GUI design methodology serves the goal of being able to rapidly enter contexts, rule antecedents, and rule consequents for processing by the rule-based KASER expert system. This enables it to be most effectively used by a single user or a team of analysts.

The problem addressed by this methodology pertains to the selection of semantic (normalized) phrases based on natural language conceptual specification for use in the loading of a context for a KASER decision support system. A corrective action may similarly be specified for use in training the KASER. There will typically be far too many phrases to enable the efficient linear (lexicographic) search through them for a semantic match. This methodology addresses the problem of how to rapidly retrieve the desired semantic phrases in real-time for contextual specification. At the same time, semantic uniformity enables creativity in the KASER once linked to this GUI.

Rule predicates are maintained in a move-to-the-head list ordering. Antecedent predicates are features, while consequent predicates are procedures. This needs to be a learning system as follows. Of course, the algorithms need to run fast too and be able to be run on parallel hardware.

The display (see Figure 1) will consist of a PC LCD screen or a wall-sized touch screen. The left-most menu will be used to display keyword and key phrases. The menu to the right will be used to display a list of possible action phrases (AP). The fields and buttons above these menus serve to filter their contents. The context and action textboxes below these menus are iteratively defined using AP menu selections. Successive constraints are not performed on the associated menu until the associated button is clicked. The rule action is a sequence of action phrases. Maximal reuse of previously used keywords and phrases facilitates retrieval and semantic specification. Keywords and phrases are only added as necessary. Selected keywords or phrases are inserted at the point of the blinking cursor. Sub-menus will not be used. Rather, the contents of each pull-down menu will be dynamically ordered upon use to best reflect their relevance to natural language constraints and/or keywords, phrases, or even letters and/or their probability of selection based on the usage history. The possibility field and metaphorical explanation button are for future expansion at this time.

All predicates bi-directionally translate to/from a unique integer id through the use of a hash table. A predicate phrase, once created, can only be destroyed through a least-recently-used (LRU) mechanism. Once a predicate phrase is expunged, its unique integer id is to be reused. A separate hash table holds antecedent (i.e., keyword and key phrases) and consequent (i.e., action phrases) predicates. Predicates having a positive sign suffix augment the context. Similarly, such predicates having a negative sign suffix will erase from the context the exact same predicate having an implied positive sign suffix, if present. Only such signed consequents may modify the context on the next iteration. The predicate matching processes will not find positive sign suffixes when matching the context (predicates having negative sign suffixes are self-erasing). This effects truth maintenance operations (i.e., retracting or replacing assertions and conclusions that are no longer true). Of course, consequent predicates may pose questions – the answer to which will modify the context via user (or possibly procedural) interaction.

The pull-down menu on the left are ordered from most-frequently-used (MFU) to least-frequently used (LFU). New entries are inserted at the top and the LFU ones are deleted from the bottom, but only to free space as needed. While it has been shown that the method-of-transposition is more efficient than the move-to-the-front method, the latter is used to update in view of the principle of temporal locality. That is, having been recently referenced greatly increases the probability of a reference in the immediate future. A logical array-based pointer system is used for the update.

Lowercase letters are not case sensitive. Moreover, the user iteratively enters zero or more predicate substrings for an implicit conjunction. This iteratively filters the predicate list in a pull-down menu. If one states, must contain, "TNT and terror", it might list such things as, “Terrorist uses TNT to blow up...,” or “TNT found in suspected terrorist camp,” etc. These constraints act as a filter on the presented items in the pull-down lists.

This process could result in too many entries or too few in the resultant pull-down menus. Then, the only recourse the user has is to iteratively retype a different set of keywords in the hope of getting it "correct". This process is laborious and thus time-consuming and hence is deemed to be unacceptable in view of our need for rapid predicate specification. Nonetheless, when used judiciously and sparingly, this filter can be advantageous.

The phrase, "The Taliban used TNT to bring down a commercial airliner" would not be retrieved by the literal constraints, {explosives, terrorists, airplane}, though clearly it should be. Another associative recall would be, "A shaped charge was dropped on a tank and exposed the populace to shock and awe". Here, "shaped charge" derives from explosives, "shock and awe" derives from terrorists, and "dropped" weakly derives from airplanes. Note that these derivations must be learned from use - not a preloaded dictionary.

The problem with generalization is that, beyond a single predicate, it rapidly loses validity through the generation of improper combinations.

3. THE GUI DESIGN

In this section, the details of the GUI system (Figure 1) are described along with the expected actions. The fundamental principle underpinning the GUI is that of facilitating associative recall. Here, the left-hand-side of the top three buttons pertains to the Contextual Keywords and Phrases (CKP) menu while, the right-hand-side of these buttons pertains to the Action Phrases (AP) menu.

All contextual and action predicates bi-directionally translate to/from a unique integer id through the use of separate hash tables (associative memories). Hash table load factors are kept below 50 percent to minimize collisions [6]. Whenever a hash table load factor equals or exceeds 50 percent, the bottommost 5 percent of the KASER rules will be expunged, the frequency use counts and (reverse) hash tables for the involved CKP and AP predicates updated, and all resulting unreferenced or "dangling" predicates be likewise expunged.

The percentage set here for block deletion (garbage collection) will be large enough to prevent thrashing, while small enough to preserve as much of the KASER knowledge base as practical. Open addressing hash tables store the records directly within the array, where in double hashing, the interval between probes is computed by a second hash function. Double hashing has a considerable advantage over linear probing. Note that if the hash table stores large records of about five or more words per record, chaining uses less memory than open addressing. Chaining, unlike open addressing, requires extra indirection for external storage. In summary, double hashing is to be preferred to direct chaining where the records are small enough to preclude the use of external storage. Again, the design of an associative GUI supports the principle of reuse.

One hash table holds antecedent (i.e., Contextual Keywords and Phrases (CKP)) and the other consequent (i.e., Action Phrases (AP)) predicates. Clearly, double hashing is used for the CKP menu and if the abstract is included in the hash, then direct chaining is the appropriate hash method for the AP menu, when RAM is limited. As mentioned before, once a predicate phrase is expunged from the KASER rule base and hence the GUI; its unique integer id is to be reused (e.g., through the use of a stack mechanism).

A random (direct) access Java database will be used to incrementally maintain updates to data structures stored in RAM on secondary memory (e.g., for reloading). This may not significantly slow down processes executing in RAM and only updates during detected idle periods and just prior to system shutdown, as necessary.

The words *INS* and *ERA* are reserved and result in automatic modification of the context textbox with the word or phrase that follows (these reserved words are delimited by a space and the phrase that follows is delimited by a comma), which enables subsequent inferences to be automatically made. These two reserved words are hard coded and are always the first two words in the initial AP menu by default.

Predicates prefixed by *INS* will augment the context. Similarly, such predicates prefixed by *ERA* will erase from the context the matching predicate, if present. Only such prefaced consequents will automatically modify the context on the next iteration of the inference engine. This process of insertion and erasure effects truth maintenance operations (i.e., iteratively retracting or replacing assertions and conclusions that are no longer valid as a result of rule actions). Of course, consequent predicates may also pose questions – the answer to which will modify the context via user (or in theory procedural) interaction.

A single AP consequent may specify an arbitrary number of *INS* and *ERA* commands which will be executed in sequential order from left to right. The context may not contain redundant integers, since it is a set.

For example, the contextual set placed in numerical order to facilitate search operations (which use the bisection search) might be, {1, 34, 35, 41, 897}. Next, a fired rule action might be: *ERA* suspect is a terrorist. Here, the quoted phrase is taken from the AP menu. If this phrase had been hashed to the integer say, 41, then *ERA* 41 will change the context to, {1, 34, 35, 897}. It is permissible to attempt to erase an integer not present. This will simply result in an "identity" operation with no messages produced. The use of the *INS* reserved word is similar.

The specification of the Context and Action textboxes are performed through selection from (insertion into) the CKP and AP menus, respectively. Direct keyboard entry into the Context or Action textboxes is never permitted and will result in a dead key.

The reason for this strict requirement to go through the menus to enter context and/or action text is to maximize reuse, thereby enabling the KASER system's creativity. The Context and Action textboxes scroll horizontally to accommodate any length entry. The CKP and AP menus similarly scroll horizontally as well as vertically (see Figure 2). They are separated by a divider max bar, which allows the menu in use to expand and cover the one not in use. This serves to facilitate the viewing of the longer phrases in either menu.

Context Undo and Action Undo Buttons

The (sub) AP list is not sorted because its MFU-ordering is deemed to be more useful. The Context Undo button deletes one integer conjunct at a time (where the CKP menu entry is defined by the CKP integer hash table) from right to left. The Action Undo button deletes one concatenated integer Action Phrase at a time (where the AP menu entry is defined by the AP integer hash table) from right to left. These operations can result in dangling menu entries, which are deleted if they were just created for this insertion and thus are not already present in the rule base.

Clear Button and Clear Operations

The topmost Clear button clears the top six textboxes when clicked. One can also backspace one character at a time within the top six textboxes to selectively clear them.

Entering text on either side of the Add, Conceptual Constraints, or Literal Constraints buttons automatically clears all text from the corresponding opposite side of those buttons. The eight textboxes scroll horizontally to accommodate any length entry.

The lowermost Clear button clears the Context and Action textboxes as well as the Possibility metric when clicked. One cannot backspace one character at a time within a Context or Action textbox to clear it. The system enforces this constraint using a dead key. Any attempt to directly enter text in either of these two textboxes is locked. The Undo buttons must be used for the purpose of deletion. In this manner, possible fragmentation of the contents of the Context and Action textboxes is prevented.

Add Button

We note that the textual entries associated with the Add and Conceptual Constraints buttons are preprocessed by a commercial, royalty-free (Java-based) spell and grammar checker prior to menu insertion.

The Add button is used to make an entry into the CKP or AP menu, as appropriate, but, ideally only as semantically necessary (i.e., when a word or phrase having the same meaning is not already present). Syntactically duplicate entries are never permitted. In this case, the Add and Conceptual Constraint textboxes are automatically cleared. Whenever a CKP phrase is selected or successfully added to the menu, it is simultaneously appended as a conjunction (separated by commas) at the right of the Context textbox. Similarly, whenever an AP phrase is selected or successfully added to the menu, it is simultaneously concatenated, separated by commas, at the right of the Action textbox (otherwise, why add it to the AP menu?). This implies that CKP and AP menu entries are checked at the time of their creation to be sure that they do not contain any commas.

Every AP points to an abstract unless empty, which is entered/viewed through a pop-up textbox and again resides in RAM. This abstract is made available to a multimedia system, where it will be presented beneath the fired rule consequent in front of the appropriate image background, when the touch-screen is tapped. Note that the allowance for a concatenation of actions results in non-determinism in the induced rule base. New menu entries are tagged with one or more conceptual constraints, although not every entry need be tagged.

Again, conceptual constraints may also be added to existing menu entries. For example, “computer” and “food” are two very different conceptual constraints on “apple”. The conceptual constraint, “computer” would have been added in the 1980s subsequent to the release of the “Apple McIntosh”. Note that “computer”, but not “food” is a conceptual constraint on “Apple”. Figure 3 shows this example.

Clicking on the Add button will insert the associated text into the appropriate menu if it is not already present there. Otherwise, this button will operate the same as if the existing entry had been selected from the appropriate menu. Note that “may” and “May” and the like are distinct case-sensitive entries.

If the Conceptual Constraint textbox below it is not empty and contains new title(s) delimited by commas, but not spaces to allow for the inclusion of phrases, then they will be added to the hash and reverse hash of the text associated with the Add button, as necessary. That is, titles not presently hashed will be inserted into the hash and reverse hash tables. Titles manually deleted from this textbox will similarly be detected as missing and thus expunged from the appropriate hash table. They will be expunged from the appropriate reverse hash table if they would otherwise become dangling pointers. Menu entries are similarly expunged when they are no longer used by any rule in the KASER rule base – as evidenced by their frequency use counts falling to zero. The deletion of a menu entry can result in a dangling conceptual constraint.

Multiple successive clicks of the Add, Conceptual Constraints, or Literal Constraints buttons will result in no further action (i.e., unless there is a change in an associated textbox). Similarly, multiple successive clicks of the Clear, Metaphorical Explanation, Save, Submit, Delete, and Help buttons will be without effect.

Conceptual Constraints Button

The Conceptual Constraints button is used to categorize and thus filter the entries in the associated menu. Again, menu entries should be tagged with conceptual constraints when added, or whenever deemed appropriate to facilitate subsequent retrieval. The relationship between menu entries and conceptual constraints is many to many. For example, the CKP entry “red” might be assigned the two conceptual constraints; namely, “colors”, “flag colors”. Hashing on “red” would bring up these two titles without repetition (i.e., redundancy). Reverse hashing on “flag colors” would bring up “blue”, “red”, and “white” (in lexicographic order). Note though that machine search is predicated on an integer ordering. Menu items may be selected with a left-click or deleted with a right-click (i.e., Windows protocols).

Alternatively, when using a touch-screen, one tap is used to select and two to delete a menu item. Two taps will bring up a confirmation box (e.g., Are you sure you want to delete, “*name*”?). In another version of the GUI, delete buttons are added at the bottom of the pull-down menus; when an entry is highlighted, the delete button is enabled. Then the user may press the delete button to delete the menu item. The delete button is then disabled.

A title is to be deleted when all of its associations are deleted and otherwise updated for each deletion – all through the efficient use of hashing. The system checks a menu entry, whenever it is updated with new conceptual constraints, to update the hash and reverse hash tables as appropriate (maintaining separate hash and reverse hash tables for the CKP and AP menus – for a total of four hash tables plus the two integer and reverse integer translation hash tables). Such compilations, unlike linked-list associations, save on runtime, which is critical to the efficient use of the GUI with scale. Reverse hashing on an unknown title will have no effect on the presented menu entries.

We use CKP for the left menu and AP for the right one. Multiple titles are implicitly OR'd and their results are presented in union in the appropriate pull-down menu.

Titles are not to be recursively treated as CKPs or APs themselves. To do so would make it impossible to specify an initial conceptual constraint. Conceptual constraints are simply added to, or expunged from, the hash and reverse hash tables, as necessary, when the Add button is clicked to enter the non-empty contents of its associated textbox. For example, the title, "car" will enable the user to find the base entry say, "1929 Porter" above that of, "Contents of Address Register (CAR)". Observe the importance of having the user be specific in the specification of titles (e.g., use "automobile" in lieu of, or at least in addition to, "car"). Note that literal constraints, if any, are independent of and operate subsequent to the effects of conceptual constraints.

Whenever an entry is selected from a menu, it will replace the contents of the appropriate textbox associated with the Add button and its conceptual constraints, if any, will be listed, in lexicographic order, immediately below. This affords the user the chance to add (and/or delete) one or more titles, which are then linked to (de-referenced from) the hash and reverse hash tables by clicking on the Add button as before. Any known titles will be quickly discovered and ignored. The number of entries produced in the appropriate menu is updated when appropriate and shown at the center-top of the menu (or twice – once at the left and once at the right, where the active menu overlays the inactive one). These integers provide the user with feedback, which supports the user in specifying better filters.

Literal Constraints Button

The literal constraints "X" followed by a comma delimiter, then a "Y" would iteratively constrain the appropriate resultant pull-down menu entries to include both case-sensitive letters "X" and "Y" in any order when the Literal Constraints button is clicked. Lowercase letters are not case sensitive. Substrings may include spaces, but not commas, which serve as delimiters. The user iteratively specifies zero or more substrings. Multiple literal constraints are implicitly AND'd and their results are presented in intersection in the appropriate pull-down menu. Literal constraints, akin to conceptual constraints in this respect, are not treated as CKPs or APs and thus are not inserted into these menus. Literal constraints are, of course, not subject to spelling or grammar checks.

Again, the literal constraining process is in addition and subsequent to that of the conceptual constraints, if used above it. Clicking on the Literal Constraints button will automatically perform a Conceptual Constraint if the corresponding title textbox of the latter is not empty and has not been previously clicked.

Possibility Feature

The Possibility is a statistic, computed by the KASER, that refers to the chance that a non-validated rule is actually valid (validated [Saved] rules are assigned a 99 percent possibility, by definition). It is cleared whenever the lower Clear button is pressed, or whenever the contents of the Context or Action textboxes changes for any reason (i.e., other than a new rule being specified [Save button], or a rule being fired upon successful return [Submit button], of course).

Explanation Button

The Metaphorical Explanation button brings up a textbox that shows the sequence (in words) of transformation rules supplied by the KASER, if any, that were applied to the context to ultimately fire the shown rule along with its Possibility. Only the context supplied by the GUI (in words) and the fired KASER rule (in words) is shown if no transforms were applied. Otherwise, the applied transforms are shown in sequence sandwiched in between these two along with their possibilities at each step, where supplied by the KASER.

Submit Button

The Submit button will send a non-empty GUI-supplied context to the KASER, which in turn will supply an action, if successful, for the user to adjudicate. Clicking on the Submit button will initially clear the Action textbox and its associated Possibility metric before forwarding to the KASER. The integer contextual set presented to the KASER is numerically sorted. The integer action sequence, received from the KASER, is to be hashed back into text for presentation to the user in the Action textbox, as previously described. Any *INS* or/and *ERA* command(s) is (are) implemented on the Context, but not shown.

Note that the set of keywords for multimedia retrieval is synonymous with the supplied context subsequent to its transformation, if transformation was used to fire a KASER rule. Notice that the supplied set of multimedia keywords may thus be a superset, which covers the fired rules antecedent. This will facilitate retrieval of the most-specific multimedia. Moreover, the multimedia system may process the fired rule consequent and its associated abstract, if supplied, to elicit further information. All available information must be included in each and every search for appropriate multimedia content (i.e., to avoid over-generalizing and thus incurring too many 'hits'). This process is to be automated and perhaps pre-saved in the multimedia system to allow for real-time selection and retrieval of multimedia content.

Save Button

The Save button saves a validated rule in the KASER rule base after checking for non-redundancy there. If the hashed, sorted, non-empty GUI context's textbox (i.e., an empty textbox here cannot be matched by definition) exactly matches an antecedent in the KASER rule base, then the

rules consequent will be overwritten with the distinct user-supplied non-empty Action textbox.

Delete Button

The Delete button deletes an exactly matching rule from the KASER rule base (there can be at most one) and reports success, or reports that the rule was not found. A successful delete will update menus, hash, and reverse hash tables (and the two bidirectional integer hash tables) as necessary to remove otherwise dangling references. This will also result in the deletion of any associated abstract for the rule consequent when the frequency use counts for these AP menu entries goes to zero.

Help Button

The Help button is simply a textual End-User How-to-Guide, which covers how to use the system. It links to an external .txt (.doc) file, which can be readily and independently updated. Indexed search (e.g., Windows Help) will be included in future development.

4. CONCLUSIONS

This paper presents the design of a system that provides the interface between a user and a cognitive system. During the learning process, the user can pose questions or supply information through the GUI.

Through this system, the learning process is accelerated, as visualization is used to provide information to the user and retrieve knowledge from the user in a symbiotic human-machine relationship.

The realization of the system interface provides well-documented benefits of diagrammatic displays for human information processing and knowledge acquisition, such as, shifts to top-down information processing strategies and enhanced recall, while alleviating adverse effects evident in confusion and motivational disengagement arising from complex diagrammatic displays.

5. REFERENCES

[1] Rubin, S. and Lee, G., "Learning Using an Information Fusion Approach", Proc. of the ISCA Int'l Conference on Intelligent and Adaptive Systems, Nice, 2004.

[2] Rubin, S. and Lee, G. "On the Use of Randomization for System of Systems (SoS) Design of Intelligent Machines", Proc. of the World Automation Congress, ISSCI, Budapest, 2006.

[3] G.J. Chaitin, "Randomness and Mathematical Proof," Sci. Amer., vol. 232, no. 5, pp. 47-52, 1975.

[4] L.A. Zadeh, "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions," IEEE Trans. Ckt. and Systems, vol. 45, no. 1, pp. 105-119, 1999.

[5] S.H. Rubin, R.J. Rush, Jr., J. Boerke, and Lj. Trajkovic, "On the Role of Informed Search in Veristic Computing," Proc. 2001 IEEE Int. Conf. Syst., Man, Cybern., pp. 2301-2308, 2001.

[6] http://www.cs.bham.ac.uk/~mhe/foundations2/node92.html.

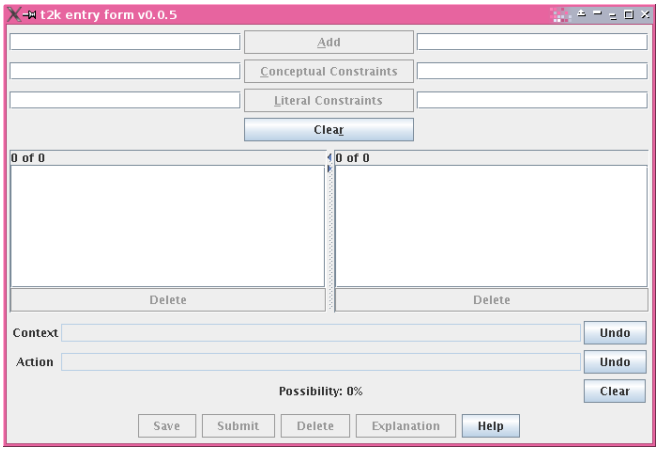

Figure 1: The GUI Configuration

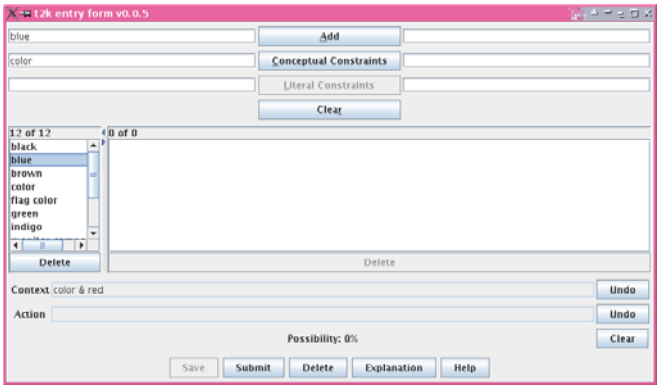

Figure 2: Example of the Scroll Feature

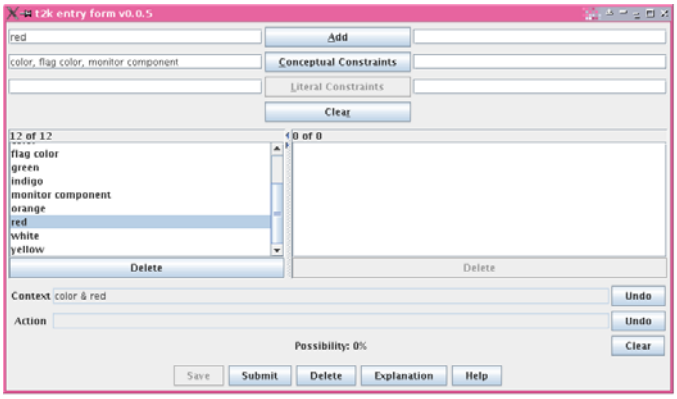

Figure 3: Example of Adding Conceptual Constraints