webAKT: Significant design, user interface and implementation issues

The original remit for webAKT specified that it should be a close-to exact replica of AKT5, at least in terms of the user interface. The reason for this was to prove that AKT5 could be re-implemented as a web application, without having endless discussions about user interface changes and enhancements. It was also proposed that, while the user interface would be re-implemented using HTML/CSS/JavaScript, in keeping with it being a web application, the underlying reasoning bits should be kept as Prolog, necessitating a communication layer between the JavaScript (on the client) and Prolog (presumably on a server).

Both high-level requirements were dropped quite early on in the project, after consultation with the AKT team. The requirement to retain a Prolog mechanism proved to be quite cumbersome, involving frequent communication between the client (the user’s computer) and the server. In addition, it turned out on closer inspection that very little of the reasoning actually made much use of Prolog’s symbolic-logic strengths, and could without much effort be handled by JavaScript data structures and programming logic. This has resulted in webAKT being a pure HTML/CSS/JavaScript application, greatly improving its maintainability and enhancements given the much greater pool of JavaScript programmers in the world.

In contrast, the steps taken to alter the user interface were incremental. It began with some very glaring legacies of software that was over 25 years old and showing its age- the most obvious example of which was the use of tables by some tools consisting of | and \_ characters to box in tables, rather than making use of a standard HTML table. This was followed over the months by a wide range of other user-interface design changes, a good number of which, especially the earlier ones, were mentioned to the AKT team. The motivations for the various changes were various, but generally arose from my intentions to bring webAKT into line with modern web-based applications, and to simplify, standardise and enhance the user experience. Should members of the AKT team wish to revert back to what they are used to from AKT5, that in most cases should be fairly simple, so my changes should be seen as experimental and suggestive rather than as cast in stone.

This note lists the most notable changes I have made. The first section addresses user interface changes and enhancements. This is followed by a section detailing internal changes (e.g. relating to the saved-knowledge-base format) which users will generally not be aware of.

# USER INTERFACE ISSUES

## A single top-level window.

In AKT5, most windows are a separate, top-level desktop window, cluttering up your computer's desktop.

In webAKT, all windows (what I call panels in webAKT) are within a single browser window. This makes it far easier and tidier to work with webAKT while doing other things on your computer.

## Separation between File and KB menus

In AKT5, the File menu only had 3, rarely-used commands, while the KB menu was overloaded with commands for both loading and saving knowledge bases as well as viewing the various components of the current knowledge base.

Conventionally, the file menu is used for working with files for the application’s document(s), while the next menu (typically Edit) is used when working with the currently-open document. I have tried to bring webAKT more in line with conventional practice.

File menu: Working with files containing a KB.

KB menu: Working with a specific KB (including switching between KBs).

## Consistency in entries under KB menu

Most menu commands under the KB menu lead to a panel displaying multiple items (statements, sources, formal terms...)

This panel in turn leads to a Details panel (e.g. the Statements panel leads to a Statement Details panel).

In it pure form, this removes handy shortcuts for a selected item. E.g. Sources button in Statements panel for a selected statement, as well as in Statement Details panel. Needs to be discussed.

## Filters for Statements panel

Easy to filter by statement type, formal term, source, and topic. Instantaneous response.

## Remove Memos menu command under KB menu

Memo is really an attribute of a specific item (ststament, formal term...), hard to see why someone should want to see a list of all memos.

## Memos are now simply another property of components

In AKT5, memos are a top-level component, like statements and sources, which can then be associated with individual items.

In webAKT, all components simply have an additional property 9in addition to the component-specific properties). This is more natural, and makes it uniform across the various knowledge base components.

## Knowledge base metadata is a top-level component, not a special memo

In AKT5, all memos are a simple text block, except for the one containing information about the knowledge base (author, location etc), which has structure for the various metadata fields.

I have made the knowledge base metadata a top-level component (like statements, sources, etc). This is accessible in the KB menu, like the other components. The only difference is that there is one metadata item per knowledge base, unlike the others, for which there are multiple items.

## Images

Images were very difficult to work with in AKT5. They had to be bitmaps (who uses a bitmap these days?), and naming was tricky.

In webAKT, images can be readily added simply by providing it URL – either on the web or on one’s own PC. Users can easily take a photo in the field, upload it to e.g. Instagram, and reference it in the KB, all in a matter of minutes. Moreover, images take up no space in the KB file – all that is stored in the file is its URL. Like memos, images are simply a property of the item they are associated with – typically, an object-type formal term – but it would be easy to allow any component (e.g. a statement or a source) to have one, or more, images associated with it.

An Images menu command is retained in the KB menu, since a KB developer or user might want to see a gallery of all referenced images; and one image can be used in more than one place.

## Source iDs

In AKT5, sources do not have an atomic ID, instead being a compound term consisting of name, location, year and suffix for interviewee sources.

I consider it to be a good idea to have atomic IDs. This makes it easy to use the ID as a key into the details for each source, as is done for all the other components of a knowledge base. I also think it is a good idea to make the source reasonably meaningful for the investigator. On the other hand, I did not think it is useful to include location and year in the ID for interviewee-based sources, since in a particular study these will hardly change.

What I have settled on is to allow the investigator – the person creating the knowledge base – to provide their own unique ID for each source, with the strong recommendation that this should be done in a reasonably consistent way, and based on the name of the interviewee (for interviewee-based sources), or first author (for reference-based sources). This simplifies the coding (as contrasted with deriving the ID from other information about the source), and allows for consistency between interviewee- and reference-based sources.

## Sources - anonymising

Sources have been anonymised (using a lookup table not actually part of webAKT) using common English first names.

## Simplify access to tools under Tools menu

This is difficult to navigate, and it might not be obvious where a particular tool resides.

Instead, have a single tools-listing panel, possibly with radio buttons or checkboxes to filter list of tools in terms of the criteria captured in the submenus.

## Unify concept of "panel" across KB, Diagram and Tools sections

In webAKT, a "panel" is a generic container that performs the role of a dialogue window, a display window, or some combination of the two. It is used to display multiple items and item details under the KB menu, to display diagrams under the Diagram menu, and to display Tools under the Tools menu. It has consistent behaviour (e.g. can be dragged around, or closed with a X (Close) button). Apart from the webAKT's top menu bar, it is (with few exceptions) the only user interface element that the user sees.

Note that this is a major difference from AKT5. In AKT5, most of the panels which appear in response to KB menu commands are similar in form and function to webAKT panels, mixing dialogue and display aspects. In contrast, those that appear under the Tool menu have a strict separation between (single-option) dialogues on the one hand, and a (crude) text display of the results of using the tool on the other hand.

## Multiple panels of the same type

In AKT5,The user can have multiple panels of the same type, such as:

* two or more Statements panels (e.g. for two different KBs; or using different filters);
* two or more Statement Details panels (e.g. to enable comparison of similar statements);
* two or more panels for the same Tool.

This raises issues about controlling the proliferation of panels in the webAKT worktop. These can be addressed in various ways.

## Collapsible-tree display for hierarchies

In webAKT, hieirachies for topics and for objects are displayed as a collapsible tree (using a standard library). This makes it easy to navigate around, see sibling entries, see the whole tree at one time, etc. This is a major difference from AKT5, which only allows one to see super- and sub-items for a chosen item.

## Diagram menu

I am struggling to come up with ideas on the best menu commands for the Diagram menu.

## Statement entry

Templates/text

## Multiple applications/websites

## Options set by controls rather than single-question dialogue windows

AKT5 typically asks a series of questions before displaying something, with each question being in a single dialogue window. In webKT, I typically use controls within the main panel.

## HTML for greatly improved displays

AKT5 is basically resricted to plain text to display things, e.g. stateents, Tool results. webAKT on the other hand has the fully pallette of user-interface styling available. So tables can be displayed as proper tables, bold and colour can be used, layouts can be attractive, and even t individual terms in statements can be colour-coded to show their types.

## Proper vector diagramming and automated graph layout

## Local use, without needing internet connection

Users can simply download the web version to their own PC.

## Autocomplete for formal terms

Currently, used in diagram node definition

## Content-rich listboxes for statements, sources, formal terms etc

## Content-rich hierarchy displays for object and topic hierarchies

## Collapsible-tree displays for object and topic hierarchies

## Change to semantics of topic hierarchy supertopics

A supertopic in a topics hierarchy is one which has subtopics below it.

In AKT5, the Manual says this (Section 7.12.3):

*Topics can only be appended to a topic hierarchy if they are a subset of that topic hierarchy. In other words* ***their ‘Boolean Search String’ can only contain formal terms/sources also included in the topic hierarchy****. [my emphasis] In this case’ farmers\_crop’ and ‘farmers\_fodder’ are both subsets of ‘farmers’. If you attempt to append a topic with a ‘Boolean Search String’ which is not a subset of that of the super-topic, for instance, if we try to append ‘scientists’ as a sub-topic of ‘farmers’, a message will appear warning you that the topic is not a subset of the super-topic.*

I do not think that most people would think that this is the way that a topic hierarchy should work, and I propose an alternative. At the time of writing this (Nov 2023) my idea has not been implemented (no have I implemented the stringent criterion for subtopics indicated by the above quotation).

Let us say that the user would like to have a topic “agriculture”, which filters all relevant statements. Because this is a broad term, with various subtopics, they decide to create a topic hierarchy, “Agriculture”, whose top-level topic is “agriculture”. (As with object hierarchies, I distinguish between the name of the hierarchy and the name of the top-level node, but that is not relevant here.) This has 2 second-level topics: “crops” and “livestock”.

Do we need to provide a Boolean search expression for the topic “agriculture”? No: because its two subtopics (which could each have a Boolean expression and/or their own subtopics) do the job perfectly well. All we need to do is to concatenate the results of applying the search results for the two (or n) subtopics.

So, rather than being very prescriptive about the relationship between a supertopic and its subtopics, we find that we do not need to have a Boolean search expression for any supertopics. It could have its own search expression (in which case its results would be concatenated with the others), but it doesn’t have to.

This means that, when we create a new topic, we should not be forced to provide a Boolean search expression for it. If we don’t, and it is not a supertopic in a hierarchy, then any search based on it would return no statements. In this case, it could be flagged as being incomplete, in the same way that we can flaf statements that do not have a source.

Implementing this idea is work in progress (Nov 2023).

Knowledge base naming and file names

In AKT5, there was quite a rigid relationship between the name of a knowledge base and the name of the file it was saved in. You can only save a knowledge base with a different name from withing AKT5 itself. If you try to change the file name, or copy the file with a different name and load it back into AKT5, you get the error message: “There is a mismatch between the file name […] and the knowledge base internal name”.

WebAKT takes a much more relaxed view. The knowledge base creator can name a knowledge base, but this is simply another piece of metadata, like say its title or authors, and can be edited like any other metadata once the knowledge base has been created. The user is given the opportunity to change the default name given to knowledge base when it is first created, but this is simply to make it easier for the user to switch between knowledge bases if more than one is loaded.

In common with most (non-web) applications, the identity of a document is represented by the name of the file it is held in, and that is true of webAKT. This name may be based on the name given to the knowledge base (and it would be sensible to do this), but this is not enforced, and it could be totally different. This makes it trivially easy for the user to manage versioning: successive versions of the same knowledge base can have the same name followed by a version or date suffix, as you probably do already for e.g. Word documents.

New approach to handling synonyms

Status: Just an idea, with some preliminary but non-working code

From early on, I’ve been concerned with the asymmetrical nature of formal terms and their synonyms. Also, some synonyms can be characterised in some way – e.g. English name, Latin name, local name for a plant – and there is no mechanism in AKT5 for handling this (apart from some sort of ordering of synonyms, which is not satisfactory.

The solution, I think, is to give all synonyms and the original term the same status. Then, when a term is used in a statement, any of the synonyms could be used in its place; or all synonyms could be displayed; or the used could indicate which type of synonym they want to see in all displayed statements, e.g. the English name. Internally, all formal terms in statements could be replaced by a made-up name for the group the synonyms belong to.

## Images

AKT5 provided a mechanism for incorporating images into the knowledge base, but this was very cumbersome. It also required to use the bitmap (.bmp) format, which is now rarely used.

Here are the instructions from the AKT Manual, Chapter 11, “Incorporating pictures and diagrams into the knowledge base”:

In order to incorporate diagrams and pictures into a knowledge base, you must adhere to the following procedures:

i) All the diagrams and pictures that are to be incorporated into the knowledge base must be in the form of bitmap files.

ii) All these bitmap files must be in the same folder as the knowledge base.

iii) The thumbnail picture or diagram to go on the Welcome Page must have exactly the same name as the knowledge base - in the above example atwima.bmp for the knowledge base atwima.kb

iv) All the other pictures and diagrams must contain the name of the knowledge base somewhere within the file name e.g. indigenous people atwima.bmp; settler people atwima.bmp

v) These file names will then automatically appear when you select Pictures/Diagrams in the second part of the Welcome Page.

In webAKT, these very strict rules have been replaced by an approach which is far simpler. Moreover, the images themselves are not actually incorporated into the knowledge base, which would have a major impact on its size. Rather, they live on the web, and the only storage requirements for the knowledge base are to hold a caption, description and URL (i.e. the web address) for the image.

How do the knowledge base developers provide the images they want to include? If the image already exists (and is available for re-use), then all they need to do is to find the URL for the image (as described below), and enter the URL into a text box in the image\_details panel. If not (for example, the developers have taken the photo themselves), then all they need to do is to upload the photo to a photo-sharing web site – such as Flickr), find its URL, and enter this into the text box in the image\_details panel.

Finding the URL for an image on the web is easy. One simply does right-click over the image, then selects the relevant option from the menu. In Chrome, this option is “Copy image address”. This can then be pasted into the text box in the image\_details panel in webAKT.

**INTERNALS**

Users will generally be unaware of these, but they represent significant technical changes from the way things are done under-the-covers in AKT5.

**1. JSON for the KB**

Current KBs are represented in Prolog. This requires conversion into some form that the JavaScript code can actually reason with. I have therefore adopted JSON as the native format for representing the KB, since JavaScript can directly load this into its internal data structures. Moreover, any common programming language can readily read JSON, this opening the door to people process Kbs I other languages (e.g. Python).

**2. JSON for statements**

By a similar line of reasoning, the structure of individual statements is also represented in JSON, as nested lists which directly mimic the nested structure of the Prolog syntax. I still use the Prolog syntax for displaying statements, and allowing users to enter new statements, but this is retained only because it is slightly more readable, and for legacy reasons.

**3. Widgets**

webAKT is highly modular, with much of the code being on widgets. A widget is a combination of JavaScript, HTML and CSS which is fully stand-alone, and shows itself as a panel (or strictly, a type of panel) in the webAKT workspace. Widgets largely are equivalent to AKT5’s Tools, with some being built in (“System Tools”), but with others being developed by users with access to JavaScript expertise (“User Tools”), with the possibility of adding these to webAKT with no involvement of a cetral authority.

**4. Scripts for loading KBs**

Although a KB is a JSON structure (see above), which is essentially a text file, the text can be wrapped up as an assignment to a single JavaScript variable, and loaded into webAKT like the other scripts it requires. This means that users can be presented with one (or more) KBs without having to load a KB from a file (either online or local).

**5. Grammar parsing**

AKT5 uses a grammar to parse statements (Table 4.2 in the Manual). Similarly, webAKT uses a library (PEGjs) which performs an identical role, and which indeed s provided with a similar grammar, almost line-for-line equivalent with AKT5’s formal grammar. This means that any future extension to webAKT’s grammar is easily added in, with no change to webAKT itself, just by extending the grammar.

**The source\_details widget**

I have made substantial changes to the way that the source-details widget works.

In AKT5, there are 2 separate dialogue windows for specifying a source – one for an interview source, and one for a reference (a publication).

In webAKT, there is just a single widget for entering a source. When the user wants to enter a new source (typically, by doing KB > Sources > New), the source\_details panel appears, with the form defaulting to an interview source (Person). The user can change this using the Source Type drop-down menu to Reference, and thereafter switches between them when you select an alternative source type. When they do this, the form changes, corresponding to the type of inputs required for each source type.

This design makes it very easy to enter multiple sources, potentially of different types (Person or Reference) without having to leave the source\_details panel, simply by clicking the Update button when each one has been completed. On clicking the Update button, the newsource is added to the knowledge base. Also, it immediately appears in the sources panel (if this is open), confirming to the user that the new source has indeed been added.

**The source\_details wizard**

When you are learning to use webAKT, it is very tedious to enter significant amounts of text – like a statement or information about a source. I have therefore provided a Wizard to provide representative entries for these two knowledge base components. Here I describe the one for entering sample new sources.

If you open up the source\_details panel to enter details for a new source, then you will see a Wizard button. Clicking on this creates a sample entry for the currently-selected source type (as indicated by the Source Type drop-down menu). This is exactly as if you had entered the information yourself. If you cick the Wizard button again, then a new sampel source is displayed, and this cycles through until you get to the end of the sample sources provided, in which case it goes nack to the first one.

Currently (December 2023) the sample sources are based on those used in the webAKT Tutorial 2, for references, and on a few sources from the Atwima knowledge base, for Person (interview) sources.

This feature was added for my own benefit during development, to simplify testing, but was left in to help people learning how to create new knowledge bases.