LEGEND User Guide

PoCs: Jeff Rye
mailto:jrye@sift.net
Dan Bryce
mailto:dbryce@sift.net
Robert P. Goldman
mailto:rpgoldman@sift.net

2025/10/24

This document describes SIFT's system, LEGEND, a knowledge acquisition (KA) tool for a subset of Ergo/Rulelog. LEGEND is a web application: users will access the tool through dedicated URLs during the initial period of experimental use. LEGEND supports KA aimed at generating two kinds of Ergo content: (1) Rules, currently limited to a predicate dialect of Rulelog (see below) and (2) Frame-based specifications of instances of known types.

The version delivered 2025/10/15 is an MWS and has a number of limitations that we describe in this document.

1 Accessing LEGEND

Users will be given individual URLs to reach LEGEND. These URLs should *generally* not be shared, but a single KA team may find it advantageous to share a single URL in order to share a single, consistent model. In the future it may be possible to provide collaboration support. URLs will be supplied separately.

2 Data Preparation

Background knowledge Because LEGEND is a web application, background knowledge files must be modified before they can be added to the application. The application does not have access to the user's filesystem, so constructs like `#include` directives will not work correctly.

1. **All include directives must be removed** from the files before upload. Instead of using include directives, upload the files to be included together with the files that include them.

2. Loading files into modules does not work properly at the moment. To the best of our knowledge, the only way to get Ergo code into a particular module is through the file

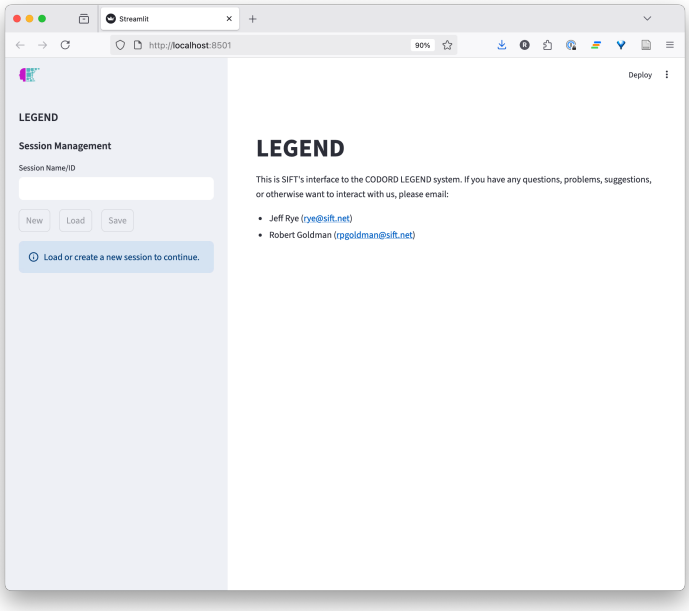

Figure 1: Initial LEGEND screen.

loading interface – there is no way for a file to mark itself as destined for a particular module. For now, **all the background knowledge files must be loaded into the same module**.

This module is not specified by the user: it will be controlled by LEGEND. However, it will be possible to load the rules derived from KA into any module of the user's choice for deployment.

Rule and Frame prompts As Benjamin pointed out at the PI meeting, it is helpful to extract, and possibly rephrase, statements describing rules and frame instances from running text ahead of time. This will speed interaction with LEGEND.

Delivery package Included as an attachment to the LEGEND delivery package is a set of modified files for the two example cases we have received, together with example prompts extracted from text.

3 Opening the application

On startup, the application will open a browser window, which will show only a message and the session configuration screen (Figure 1). The application window will be largely blank until one creates or reloads a session.

4 Starting a Session

The KA *session* is the organizing container for KA on a particular subject. To create a session, type a session name into the textbox in the left tab, and press the "New" button.

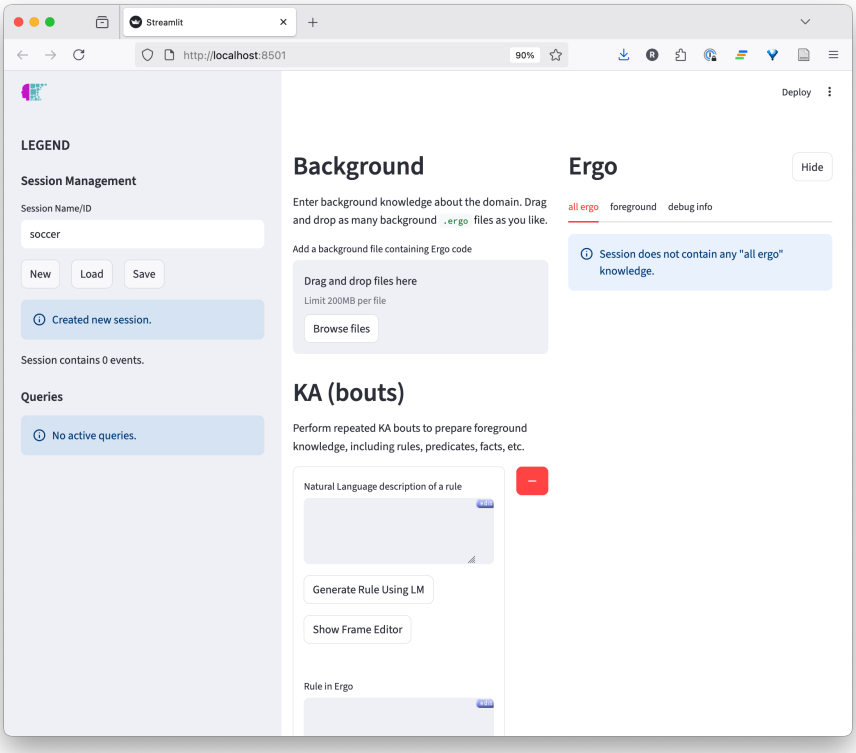

Figure 2: LEGEND screen with empty session.

After creating a session, LEGEND will reconfigure to a "triptych" display (Figure 2). The leftmost pane contains session management controls, the middle pane is where the KA takes place, and the rightmost shows the Ergo code that is available, both "background" (given) and "foreground" (the results of KA).

5 Adding background knowledge

The next step in starting KA is to supply any required background (given) knowledge. To do this, you may either drag and drop files into the upload area or use a file picker. See Figure 3, which shows the file "`soccer_team_background.ergo`" loaded. You may load multiple background files, as needed.

The right pane of the window will update to show the uploaded background knowledge (this pane can be collapsed to allow more space for KA and session control).

Files can be deleted from the set of background knowledge using the minus sign in the bright red box. Again, the right pane display will update accordingly.

Unfortunately, users cannot *update* background knowledge files, because of the limitations of a web application (the web application has no access to the user's file system), so all one can do in that case is delete a file and re-upload it.

After successfully uploading the background knowledge you want, it's a good idea to save the session. At present, the session will be saved on the server. You can reload it by name.

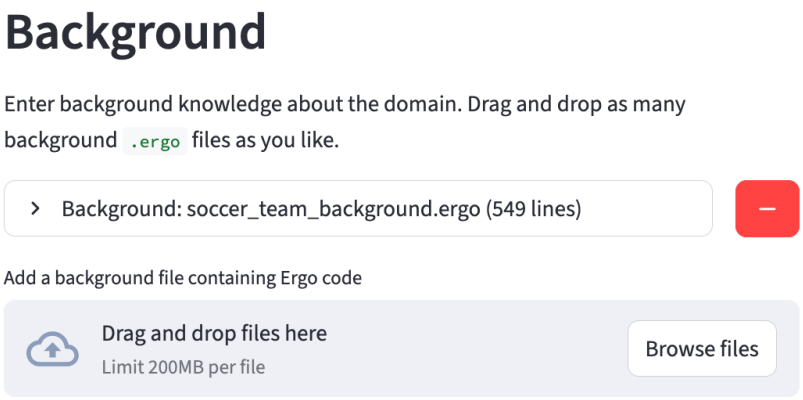

Figure 3: Background file selection.

6 KA in bouts

KA in LEGEND proceeds in what we are calling "bouts." Each bout centers around a piece of textual input, which you paste into a text box (Figure 4), and then proceeds either to rule authoring (Section 7) or frame population (Section 8).

Note: "bout" may not have been the best term for us to use, since it implies strict ordering. One can enter the NL for a bout, realize that it's not one you are ready to work on, and postpone KA in favor of another. For example, if populating a frame instance, you might realize that you need another frame instance as a property filler. That's ok! Start a new bout and come back to this one.

Warning: if you save a session and restart, incomplete bouts may not properly be restored. We are working on this.

7 KA for Rules

Figure 4 gives an example of a rule to be added to the foreground knowledge. To get suggested rule(s) press the "Generate Rule using LM" button. LEGEND will assemble a prompt from the input NL and information from the background knowledge, and submit it to a LLM (right now we are using Anthropic's Claude).

The result, for the "necessary roles" prompt can be seen in Figure 5. LEGEND will display the chain of thought output from the language model and, separately, the Ergo rule(s).

As noted above, rules are currently written in predicate notation, though we hope to translate to frames soon. We use "link predicates" to make our predicate notation work, and you can see the link predicates in the foreground knowledge in the UI.

At this point, you can edit the Ergo rule, modify the NL input and retry, or add the rule to the current set of foreground knowledge. We will soon be adding the opportunity to instruct the LM to modify its proposed rule. That capability already exists in LEGEND, but has not yet been connected to the user interface.

Note that any time you are satisfied with KA products, it is a good idea to save the session, and download the products, as well (see Section 11). LEGEND will be under

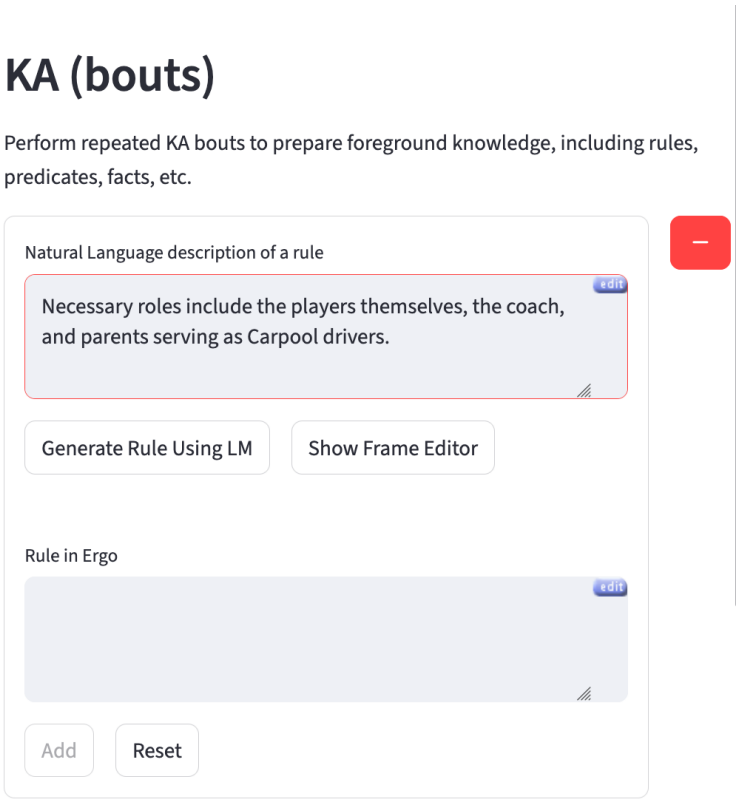

Figure 4: Entering the NL prompt for a bout.

constant development, attempting to add new features and fix bugs, over the course of the next two weeks, so save early and often! Sometimes innocent repairs to the system will render previous sessions impossible to reload.

This would also be a good time to (re)run stored queries (see Section 10), to see if adding the rule(s) has the desired effect.

8 KA for Frame Instances

Figure 6 shows what happens if you enter NL and ask for the Frame editor, instead of rule generation. In the current version of the interface, you may choose the frame type yourself. However, we now include suggestions generated by the LM: this will be particularly helpful for the test evaluation problem, which has a much a larger set of types from which to choose. We are also working to integrate LM-based generation of suggested property assignments. As with the rule generator, the frame editor displays an Ergo translation of the KA result, and this translation can be edited in place before adding the rule to the foreground knowledge.

This figure shows an example of interleaving bouts. We would like to be able to record the fact that Bill, Jr. is Bill French's son, but we haven't added Bill, Jr. to the model yet. We can add a new sentence and create Bill Jr, as shown in Figure 7.

After we have added Bill, Jr. we can go back to the previous bout, and select him as Bill French's son. See Figure 8.

As pointed out in the previous section, this would be a good time to save the session,

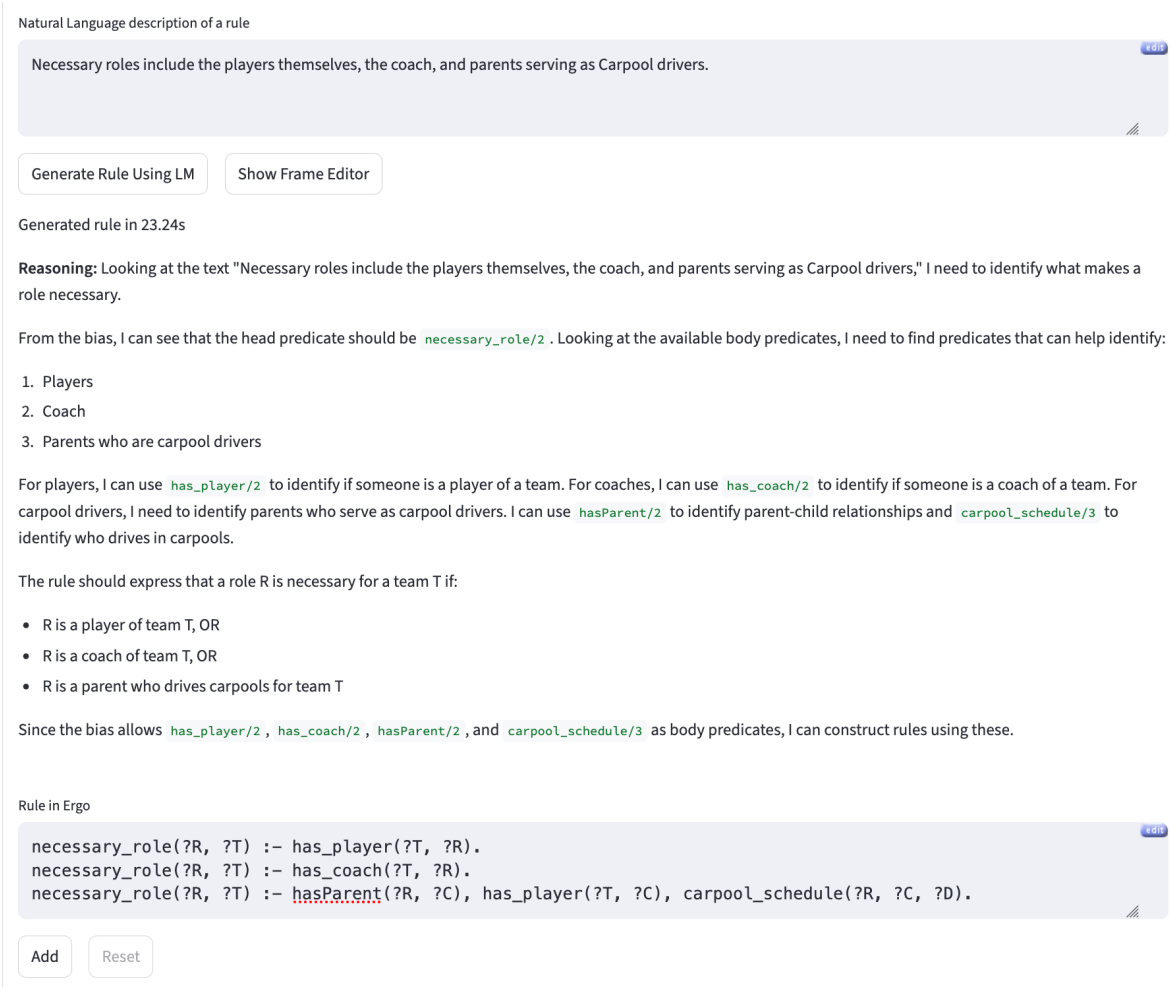

Figure 5: Rule produced by LEGEND.

and possibly the foreground Ergo code, and also to run any stored queries.

8.1 Frame Editor FAQs

1. *Is there any intelligence in the current version that should populate the table? If so how has your team achieved this with prompting, what successful prompting techniques work for you?* No, we do not have any intelligence to populate the table. We are working on an LM-based approach to populate the frame editor from NL text. Unfortunately, this is not yet reliable enough to deploy in our public LEGEND instances.

2. *Is it possible to force the generation to not use Prolog Syntax?* Not in the deployed LEGEND instances. However:

   1. The contents of the "Rule in Ergo" text field are editable. Any time the generation produces something you wish to change, you may edit the resulting Ergo as you see fit.

   When you see *strong_prohibition*, you may manually edit it to read 'prohibition(strong)'

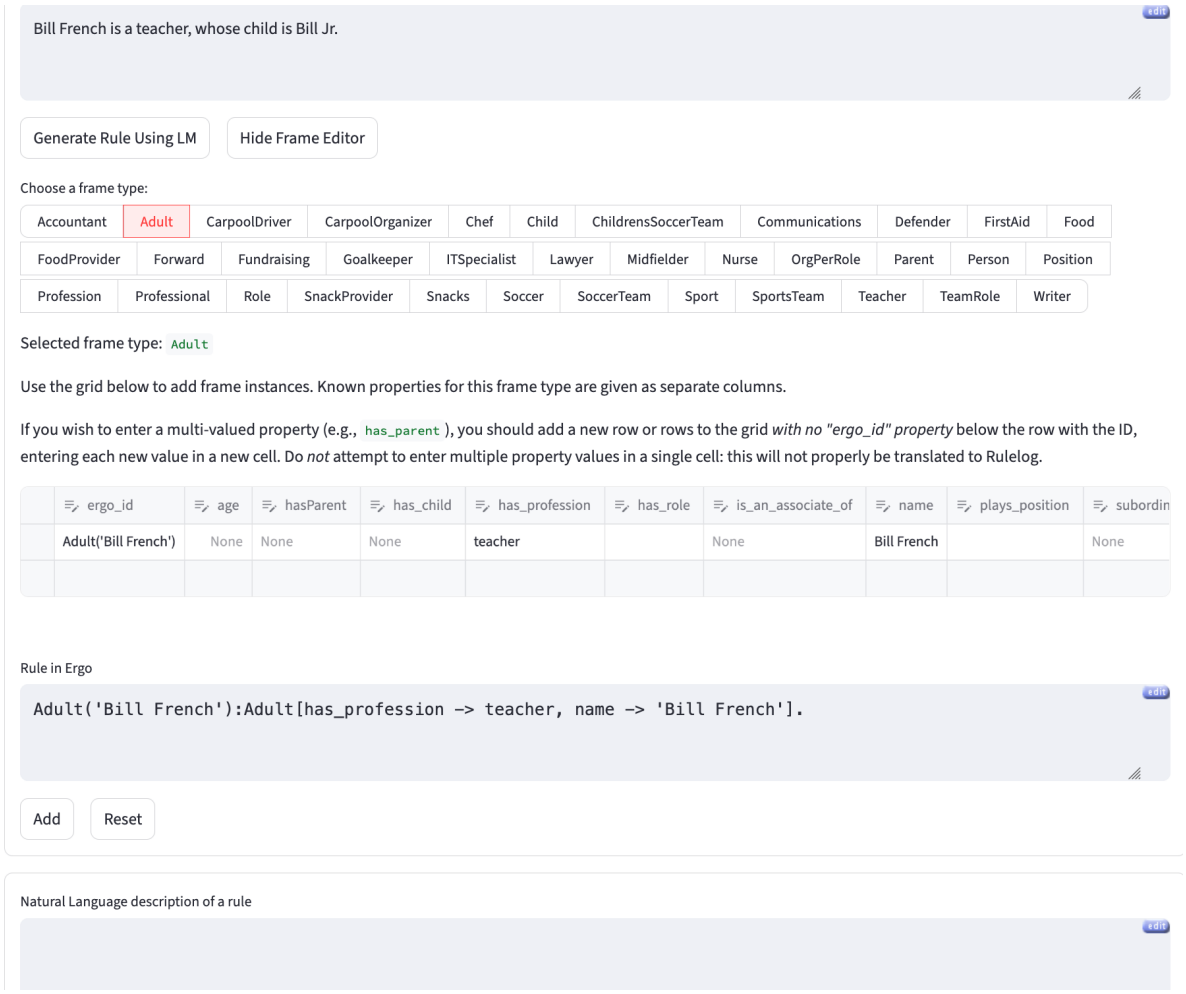

Figure 6: Adding a new frame instance.

before clicking the "Add" button to introduce the rule to the foreground knowledge base.

In the most extreme cases, you can simply write any Ergo you want in that field without even running generation with the LM.

We believe the LM speeds the transformation of NL to Ergo, but the objective is to ease the user's KA process. So please feel free to edit the results and move on when that's easier than trying to find a magic prompt to get the LM to generate something specific.

2. We have also been unable to get the LM to produce the 'prohibition(strong)' form. As a partial mitigation to this, LEGEND automatically introduces some helper rules to map *strong_prohibition* to 'prohibition(strong)' as needed when running queries.

I have attached the file containing the relevant helper Ergo. When running queries in the LEGEND interface, these rules will be present. This means that you should be able to add rules of the form *strong_prohibition* and still refer to them as 'prohibition(strong)' in your queries.

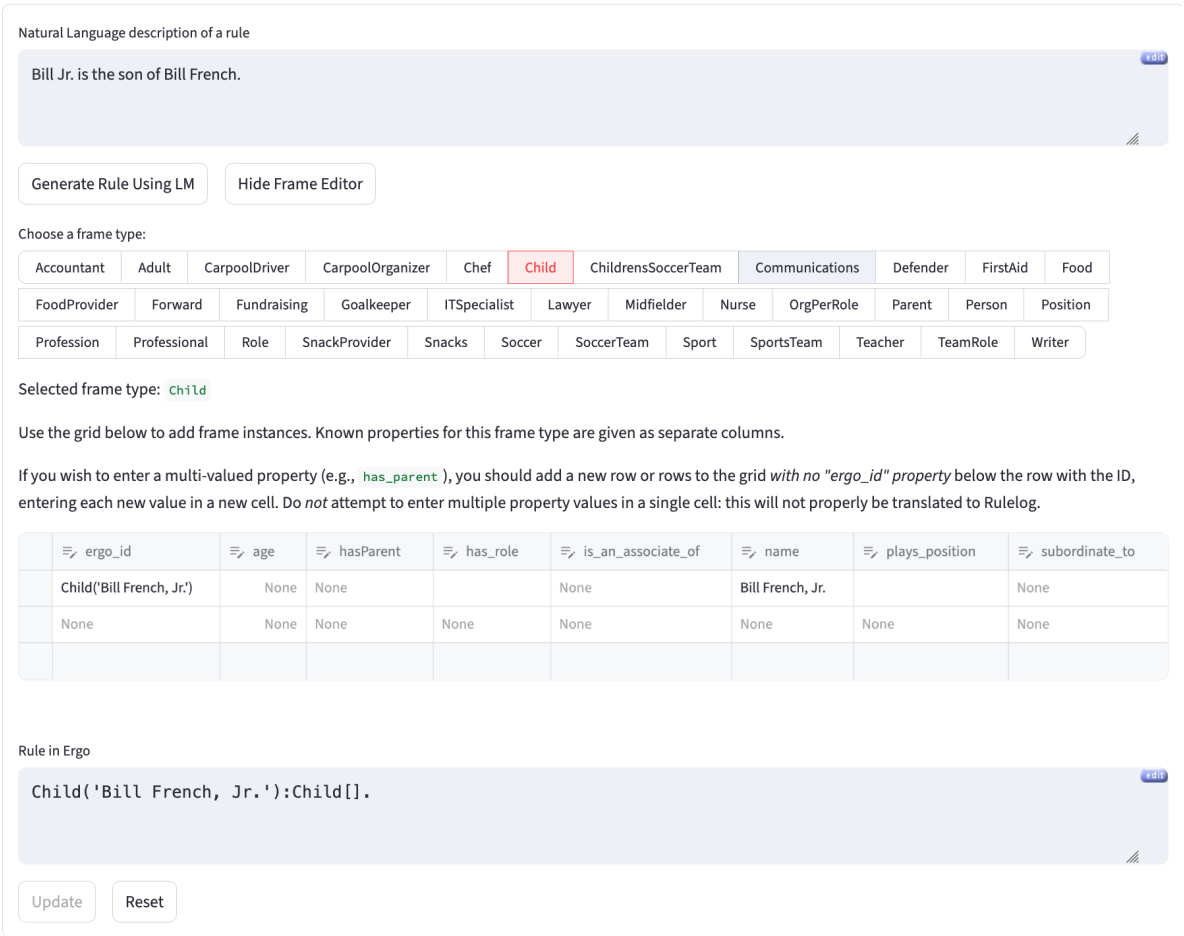

Figure 7: Adding Bill Jr.

9 KA results

At any time, you can inspect the background knowledge and KA products that have been accepted. For example, see Figure 9.

10 Queries

To help users determine whether the KA has been successful, LEGEND accommodates stored queries that can be tested against the combination of background and foreground knowledge. For example, if working on the Soccer exercise, one might want to run the `%test_scenario`. In order to run this query, one needs to add the Ergo file or files that contain the test predicate definitions to the knowledge base. For the Soccer exercise, one would add the `scenario_test.ergo` file, as seen in Figure 10.

One can then find the tests to run by filtering the knowledge base for a word, e.g., "scenario" as seen in Figure 11.

In Figure 12, we see a user adding this top-level query to LEGEND, and calling it `test all`. As this figure shows, queries are added in the middle pane of the LEGEND interface.

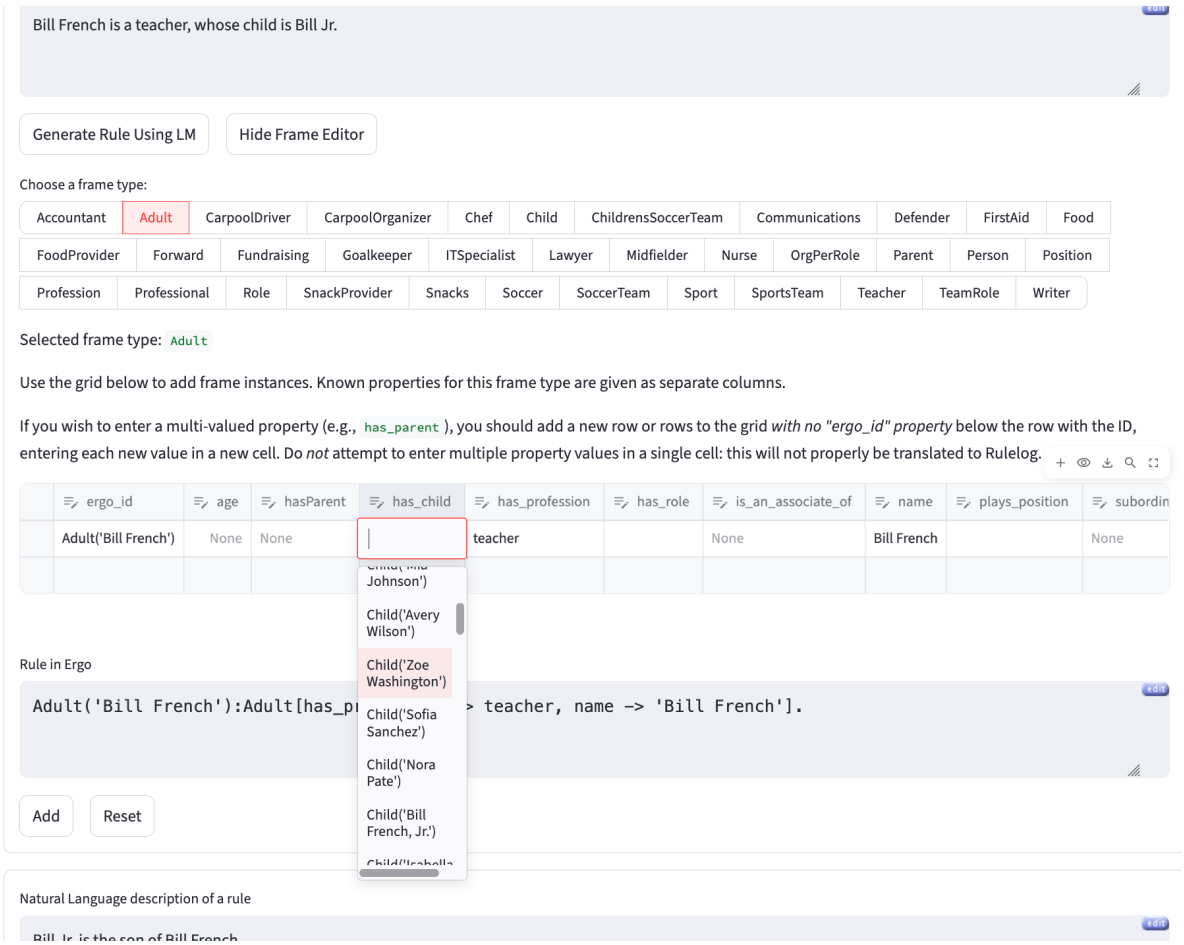

Figure 8: Adding Bill Jr. as Bill's son.

Queries added to the session can be run from the *left* pane, as shown in Figure 13.

11 Downloading KA products

Finally, it is possible to download either the full set of Ergo code from LEGEND, or just the newly-learned, foreground knowledge. This is done using the right pane, as shown in Figure 14. Of course, this can be done as many times as you would like.

12 Lessons learned

While LEGEND is not a *conventional* IDE, it is nevertheless an IDE for the Ergo language. For this reason, it is somewhat awkward to use as a web application. The problem is that programming is still very much tied to use of a filesystem, and as a web application, LEGEND is denied access to the local filesystem. This makes it awkward to work with code that references pathnames. This limitation is particularly acute for the Ergo programming language, since Ergo relies on include directives to ensure that the code in multiple files can be loaded into a coherent process image, and allocating code to different modules is inherently

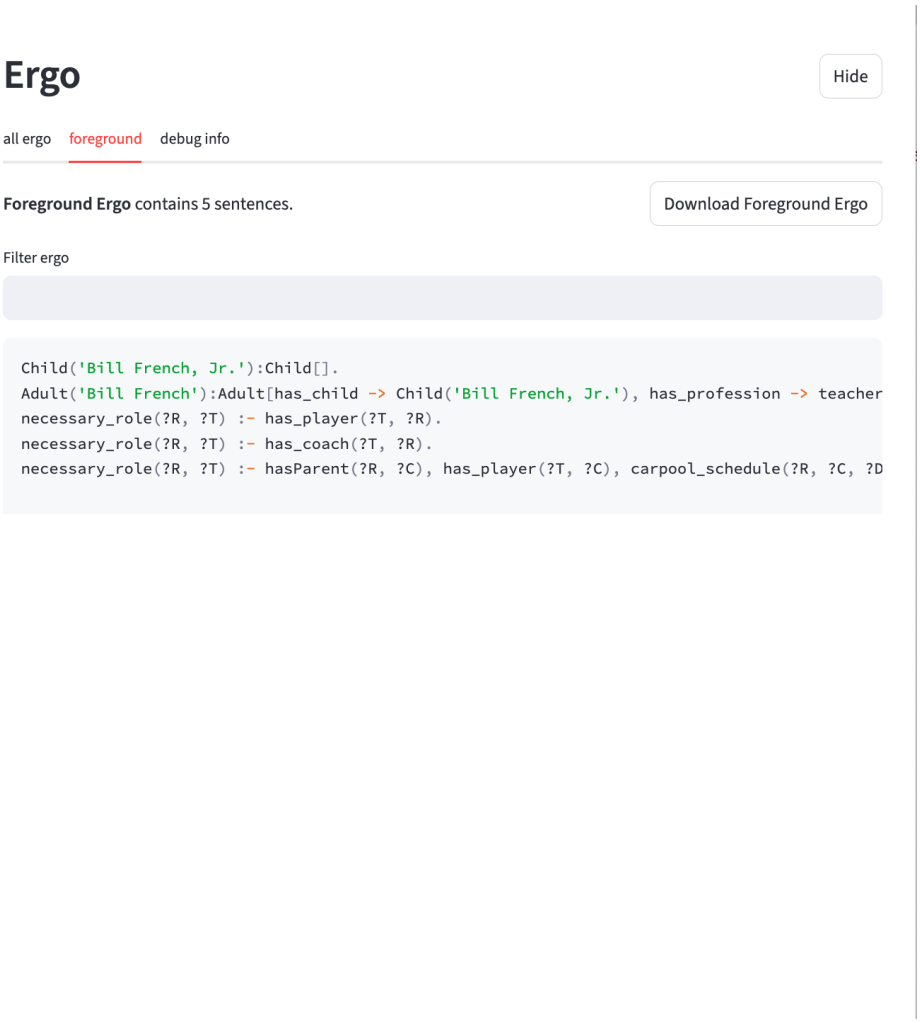

Figure 9: Accumulated foreground knowledge, showing rules and frame instances.

tied to file loading: in Ergo files are assigned to namespaces when they are loaded. Ergo does not allow the programmer to assign a namespace to their code in the file itself, unlike many other programming languages.

There are offsetting advantages to deploying LEGEND as a web application at this stage of development, however. For one thing, in this early stage of development, it is a great advantage to be able to rapidly incorporate fixes into the web application, without the need for an update delivery process. We can also collect information about the system's use that will guide software maintenance and enhancement. These advantages currently overwhelm the disadvantages, but eventually we would like to see LEGEND move to being a desktop application.

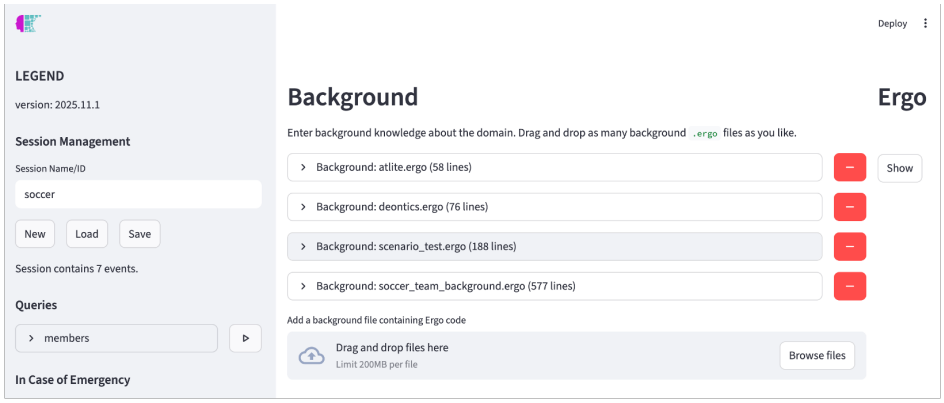

Figure 10: The test predicates file (scenario_test.ergo) from the soccer team warmup exercise is highlighted after being added through the "Drag and drop files" option shown below the file names.

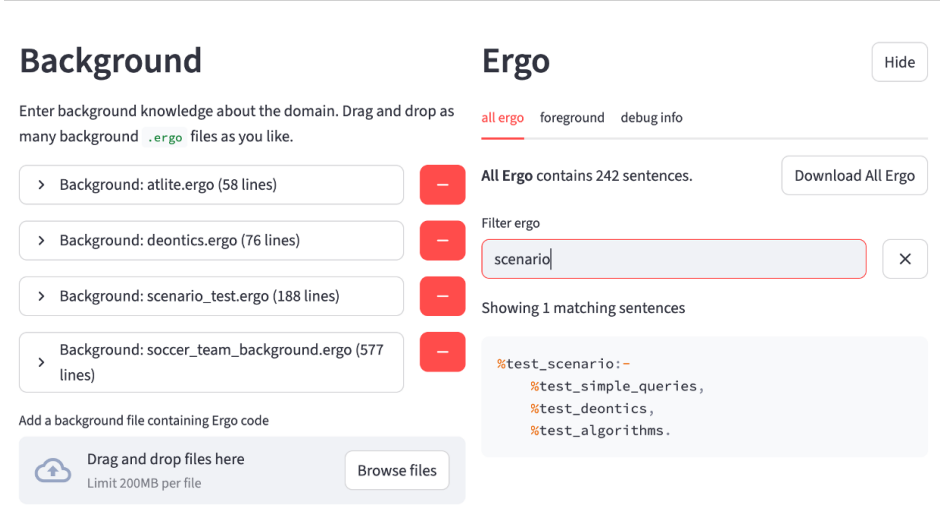

Figure 11: Finding the test queries from the test predicates file.

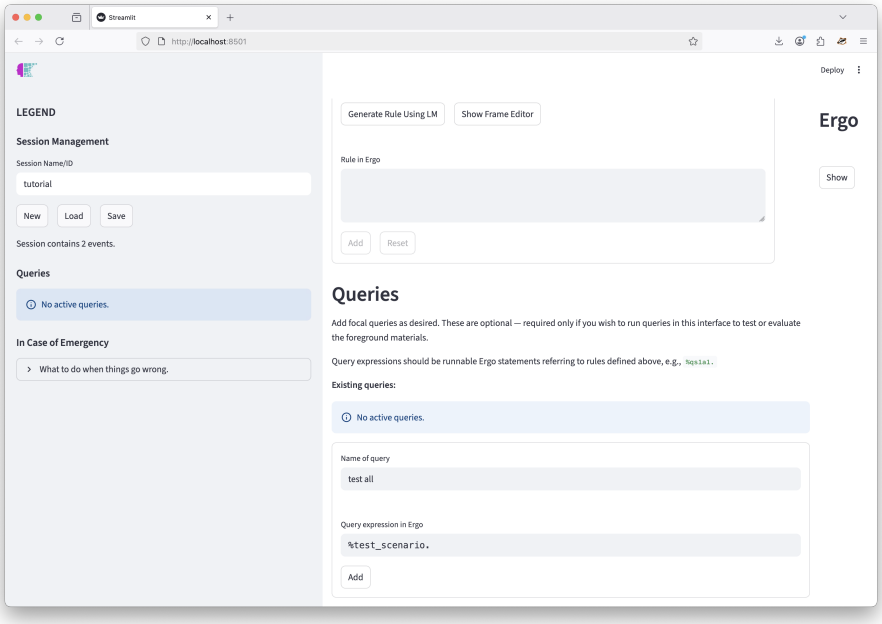

Figure 12: Adding a query from the soccer team warmup exercise.

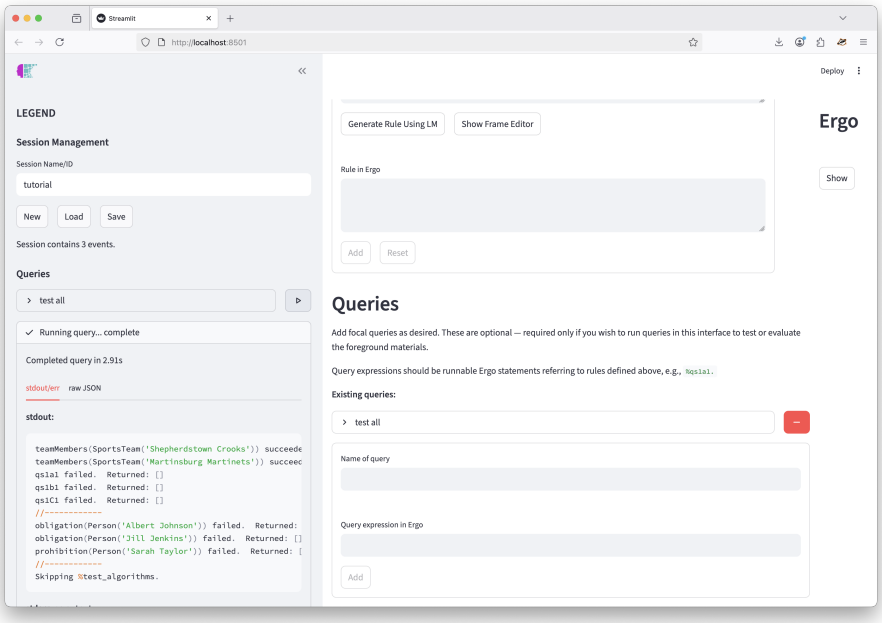

Figure 13: Running a query from the soccer team warmup exercise.

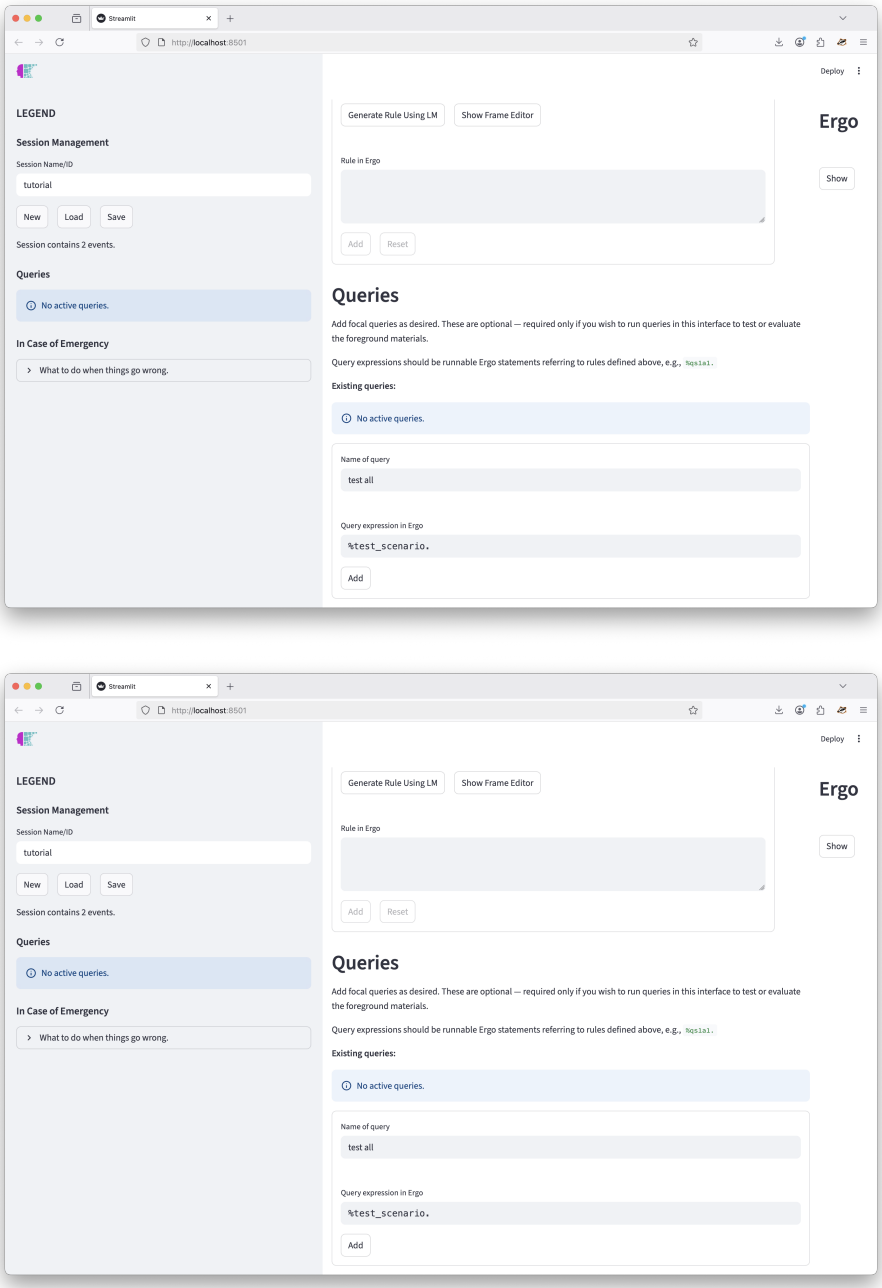

Figure 14: Browsing and downloading Ergo code.