EURISKO

Tom Rochette <tom.rochette@coreteks.org>

December 21, 2025 — 77e1b28a

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Where should code be stored? Should it be part of the concepts?
  - If it can be stored in files, then we can put it under version control
- One current limitation is that algorithms and definitions are "hardcoded" in the sense that they are provided to EURISKO and not manipulated by EURISKO itself. Given that we may have thousands or more concepts, and that concepts may have tens to hundreds or thousands of slots which generally contain such algorithms/definitions, writing code for each of them will be time consuming while it could be done by the AGI itself
- Can heuristics specialize/generalize (that is, move from one concept to another in a hierarchy chain)?
- Concepts can be see as classes in OOP, where generalizations and specializations are basically the supertypes and subtypes

0.4 Nice to have

- Definition editor
- Be able to revert changes in the logic (support some form of command pattern?)

1 Overview

1.1 High level overview

- EURISKO uses an agenda-based approaches where different task are part of a list and where priorities change over time
- The regular workflow is as follow:
  1. Pick the highest priority task in the agenda
  2. Work on the task (this may generate new tasks)
  3. Evaluate how "valuable" working on this task was (using evaluation criteria)
  4. Return to step 1.

1.2 Notes

- GET/PUT functions to retrieve/update data
- Calls to GET should specify the reason behind the call (Existence, Length, Some, Up-to-date) and how much resources can be spent (Time, Cells, Queries)
- Data has slots (attributes/fields/properties)
- Data is mostly key/value

- Data types are basic types (int, float, double, string, character, boolean, date, date-time) or abstract types (set, bag, list)
- The control structure is part of the data
- The concept of agenda is used to track topics of interest
- New agendas are generated when existing agendas are too big and agendas are merge together when they become too small
- Space and time bounds are computed for the heuristic that will be executed
- When a task is proposed which deals with a concept C, EURISKO ripples up from C along the Generalizations links looking for topics, halting as soon as it finds one
  - A pointer to the task is put on the agenda of each topic encountered
  - There can be several pointers to the same task simultaneously existing on different agendae
- Worth is capped to 999
- **Heuristics are stored in their particular concept**
  - As concepts are hierarchically defined, the descendants of a concept can make use of their ancestors heuristics in order to attempt to solve problems

2 Basic main loop

- select and work on a topic
  - given a topic, select and work on a promising task
    - given a task, select and obey a relevant individual heuristic rule
    - execute a post-mortem of the task execution
- Selecting a task is done as follows
  - The top task's reasons are evaluated carefully, and its rating is updated
  - If, after reevaluation, the top task's rating falls below that of task number 2, we merge it back into the agenda, and repeat this step
  - Some task will stay at the top of the agenda and be elected for execution

3 Tasks

3.1 Agenda maintenance

- Merge agendas
- Split agendas
- Select agenda

3.2 Task execution

- Evaluate a task priority reasons (why is it the top priority task and should it still be?)
- Select task
- Selecting rules which may (help to) satisfy the task
- Compute time and space bounds
- Execute task
- Post-mortem task
- Generate task

3.3 Generic

- Keep the KB consistent

  - Create new concepts which do not exist but are listed in other concepts
- Toggle when to compute/cache results for a concept

4 Post-mortem

- Task execution is monitored for duration, resource consumption, rules success, etc.
- Decisions are logged for further review
- New units created are recorded
- How many time a slot was used

5 Implementation

- Stored data
  - Data we are reflecting on (data)
  - Data about our reflecting (meta-data)
- Textual format
  - Either plain text, json/yaml-like or xml documents would be appropriate
  - It's important that the format be flexible
- Could be represented in a database using two tables
  - Concept table, mostly an ID and a name
  - KeyValue table, which contains key-value for each concept
  - For example, a concept EnergyGun would have an entry ID = 1, Name = EnergyGun in the Concept table and many entries in the KeyValue table
    - ConceptID = 1, Key = Generalizations, Value = Anything
    - ConceptID = 1, Key = Generalizations, Value = Weapon
  - The database could also be designed to have the keys as part of a third table, such that
    - Concept <-> KeyValue <-> Key
    - Where KeyValue has ConceptID, KeyID and Value and Key has ID, Key
  - The advantage of using a database which uses an SQL language is that it provides tools to do rapid querying over all concepts
- Each slot should describe what it means to have a value stored on itself
  - type
  - constraints (length, min/max, regex, etc.)

6 Slots/Facets

- Name: What shall we call C when communicating with the user?
- Generalizations: Which other concepts have less restrictive definitions than C?
- Specializations: Which concepts satisfy C's definition plus some additional constraints?
- Examples: What are some things that satisfy C's definition?
- IsA: Which concepts' definitions does C itself satisfy?
- InDomainOf: Which operations can be performed on C's?
- InRangeOf: Which operations result in values which are C's?
- Views: How can we view some other kind of entity as if it were a C?
- Intuitions: What is an abstract, analogic representation for C? **(removed)**
- Analogies: Are there similar (though formally unrelated) concepts?
- Conjectures: What are some potential theorems involving C?
- Definitions: How can we tell if x is an example of C?
- Algorithms: How can we execute the operation C on a given argument?
- Domain/Range: What kinds of arguments can operation C be executed on? What kinds of values will it return?
- Worth: How valuable is C? (overall, aesthetic, utility, etc.)

- Interestingness: What features make a C especially interesting?

6.1 Subfacets

Contains heuristics for dealing with that facet of C

- FillIn: How can entries on this facet be filled in? These heuristics get called on when the current task is "Fill in facet F of concept X", where X is a C
- Check: How can potential entries on this facet be checked and patched up?
- Suggest: If we get bogged down, what are some new tasks (related to this facet) that we might consider?

7 Heuristics

- If a concept is composed of many components, then get rid of a component and create a new concept out of it (generalization/abstraction)

8 See also

9 References