The Free Life Planner

(DRAFT - Do NOT Distribute!!):

A Virtual Secondary Social Safety Net

Andrew John Dougherty
FRDCSA Project

Abstract

Could free software artificial intelligence be uniquely positioned to help alleviate poverty, hunger, disease, and a slew of other aptly-termed "wicked problems?" There are a number of positive indicators. Free software such as Linux has the benevolent property that it may be copied ad infinitum for essentially zero cost, enabling world-wide distribution. Moreover, modern artificial intelligence software on commodity hardware is capable of solving an increasingly wide range of problems, often with superhuman performance. Additionally, the digital divide is disappearing, with intermediaries able to reach the rest. These factors mean that essentially everyone may soon have access to or benefit from free superhuman intelligence. If so, a virtual secondary social safety net system could be developed (at near-zero development and marginal cost), which could help people to be more self-reliant and less dependent on a primary social safety net. This could help to improve quality of life, reduce suffering, and save lives. We propose the Free Life Planner (FLP), a systematic life-planning system intended to help fulfill this opportunity.

Introduction

Purpose of the Software

To End the "Information Dark Ages". The late Dr. Jaime Carbonell had the praiseworthy goal of "getting the right information to the right people at the right time in the right language in the right medium with the right level of detail."[(27)] Anywhere this goal remains unsatisfied we consider to belong by definition to what we term the "information dark ages."

The author recently heard of a situation in which a patient with a pacemaker, whose medical records had not been transferred to the appropriate physician, was scheduled to undergo medical treatment involving electrical shocks. This might have proven serious or fatal, had not a confidant serendipitously discovered the issue.

Such situations underscore several facts: that some people still live in the "information dark ages," that some people must self-advocate, that life-and-death matters are best not left to chance, and that systematic or catch-all solutions are needed.

The Role of the Free Life Planner. In this paper we will outline the case for a free software life planner, and show work done to date. The Free Life Planner (FLP) relies on a variety of (mostly) planning and logic-based methods to lock the situation down and to provide relative safety guarantees. It is intended to be a systematic solution that helps to manage the logistics of daily living, to plan to achieve and maintain the conditions of life.

A central pillar of this approach is the creation of a (semantic) simulation or digital twin,[12] which is then used to essentially prove,[1] whenever feasible and with higher confidence, that bad things won't happen to people. Fortunately, many free software tools exist that can among other things prove safety-critical guarantees in diverse settings.[39] A critical task for FLP is to apply the vast corpus of existing free software[2] (AI) technologies to bridge the gap between available tools and end users to ensure last-mile delivery of these software capabilities to all consenting persons, promoting social good and enhancing welfare.

It is possible to encode facts and "rules" that in practice effectively govern real life into formats that are executable and machine-readable. When reasoning about what actions to take,[31] potential actions can be systematically proposed and vetted on the basis of these rules,[37] in these cases faster and more precisely than a person could. Because the process of rule acquisition is amenable to automation, it is possible to create large rulebases from textual, experiential and other sources. A variety of trust management mechanisms exist which may help users determine which facts and rules they feel they can trust.

Organization of this Paper

First, we consider the philosophical and technical motivations behind the project. Next, we examine FLP's capabilities by focusing mainly on three representative custom subsystems (from over one thousand). These systems were chosen for their relation to the three representative wicked problems[43] mentioned earlier: poverty, hunger and disease.

- **Financial Planner**
  - Reasons over personal finances, and is aimed first toward resolving the common cash-flow issues usually associated with a fixed or low income
- **Meal Planner**
  - A comprehensive diet management system for individuals and families, aimed at minimizing cost and effort and maximizing nutrition and palatability
- **Health Planner**
  - A set of systems to maintain health, including the ability to navigate potential or extant medical conditions through preventative, ongoing and restorative care

We then talk more briefly about other supporting features in various stages of completion.

The next section considers some possible objections which regard the scope and applicability of the software, including information security issues, "death by GPS," corruption of facts and rules, criminal misuse, legal liability, the digital divide, "no one size fits all," monoculture, software maintenance and real-world effectiveness.

We then look at the distribution of the software, and some factors that are presently limiting

[1] By invoking planners, theorem provers, formal verifiers and other algorithms

[2] We will unpack the concept of free software later in this paper

FLP (and its parent project, the FRDCSA). We next suggest future work, and finally present our conclusions.

Background

Philosophical Motivation

Because free software may be copied to or transitively minister to and hence service most everyone in the world, and because free software is a hermetically closed system in which one only gets out what one puts in, it puts pressure on developers to contribute software systems which are commensurate to the weight of the world's problems and needs.

In this section, we look at solutions to these problems and needs by considering some supportive premises, claiming some "beneficial ontological truths," and looking at conclusions. Their alignment facilitates the motivation for the FRDCSA/FLP solution concepts. The philosophy behind FLP is a special case of the humanitarian philosophy behind its parent project, the Formalized Research Database: Cluster, Study and Apply (FRDCSA)[18] weak artificial intelligence project. We present here a rough outline:

- **Premises:**
  - A) Developers can work on separate (often orthogonal) components
  - B) Separate components may be united through systems integration

- **More Premises / "Beneficial Ontological Truths:"**
  - C) A heuristic for the creation of more capable software systems: bigger is better
  - D) Software is non-rivalrous [3] [32]

- **Conclusions:**
  - 1) Large systems may be built [from A, B]
  - 2) Larger systems have the potential to be better [from C, 1]
  - 3) Software may be effortlessly replicated into different physical copies [from D]
  - 4) These integrated systems may then be copied endlessly [from 2, 3]

Premise A: Developers can work on separate components. Specialization and modularization enables developers to work on separate subsystems. It is often psychologically desirable for developers to work on separate components. Developers often begin coding solutions to specific problems they see, have different interests, wish to define their unique identity, etc.

Premise B: Separate Components May Be United Through Systems Integration. Modern workflows enable the integration of and continuous integration testing of various software components.

"Beneficial Truth C:" Bigger is Better. One heuristic for the creation of more capable software systems is that bigger is better. It should be relatively straightforward to prove,[4] [34] [5] [35] as

[3] Free software is not merely non-rivalrous, but anti-rivalrous

[4] An early but incorrect statement (without proof) of a consequence of Gödel's first incompleteness theorem

[5] An under-construction mechanization of the proof

a consequence of Gödel's first incompleteness theorem, that for any given computer program, there always exists a more powerful program.

Theorem 1 $\forall p \in TR(\exists q \in TR(Larger(q,p) \wedge Stronger(q,p)))$

A consequence of Theorem 1 is that there are infinite sequences of increasingly powerful programs. Naturally, the program lengths of the individual programs in these infinite sequences must eventually increase, because for any given program length, there are only a finite number of programs of less than or equal length.

Theorem 2 $\neg\forall p \in TR(\exists q \in TR(\neg Larger(q,p) \wedge Stronger(q,p)))$

Working backwards from Theorem 2, we can conclude that a heuristic for increased program strength is increased length. (Not every program of length *n* is going to be powerful for its length, however, some of them will. This heuristic is also independently derivable from algorithmic information theory).[8]

Given the existence of hundreds of thousands (maybe even millions) of unique free software programs, a natural way to create a very large program is to simply collect as many of these programs as possible into a very large system. Practically speaking, software conglomeration requires the creation of packages of this software, such as operating system packages (like Debian packages), or at least packages for other package managers (such as PyPI Python packages).[5]

But current conglomeration efforts practiced by most programmers fall short of a comprehensive, systematic, first-class effort. The proof that this method is not being adopted is the lack of packages for major Linux distributions of extant Linux AI systems such as OpenCyc, ConceptNet, Enju, etc. Ideally, there would be a large project like the Manhattan project or Apollo project to inaugurate AI, whose difference with current software packaging processes might be as pronounced as that between a space shuttle and a model rocket. Yet, presently there remain a large number of powerful and often high-profile free software programs which aren't immediately installable via any packaging system.

Therefore a major mission of the FRDCSA project is to package and collect as much of this software as feasible, ordered by some norm over utility, first to the packaging process itself, and then to end users. While packaging is a difficult process, it is plausible that it may be automated to an increasing degree.

FRDCSA's own efforts to automate packaging[17] are being independently superseded by those of Debian Developer Jelmer Vernooij,[4] who plans eventually to document the complete package generation process. Some prerequisites, however, already are covered.[38] [29] [40]

"Beneficial Truth D:" Software is Non-Rivalrous / Has the Zero-Marginal Cost Property. We have claimed that larger systems have the potential to be stronger. The next "beneficial ontological truth" involves the nature of information itself. We start by defining the concept of non-rivalry.

> "A good is considered non-rivalrous or non-rival if, for any level of production, the cost of providing it to a marginal (additional) individual is zero." - Wikipedia article on Rivalry[33]

The fact that software is non-rivalrous (and hence has the zero-marginal cost (ZMC) property) enables both the creation of larger systems through systems integration as well as the distribution of those systems for all possible end users.

> "That ideas should freely spread from one to another over the globe, for the moral and mutual instruction of man, and improvement of his condition, seems to have been peculiarly and benevolently designed by nature, when she made them, like fire, expansible over all space, without lessening their density in any Point, and like the air in which we breathe, move, and have our physical being, incapable of confinement or exclusive appropriation." - Thomas Jefferson(9)

Thus, a characteristic of information (such as ideas and software) is "freedom." We next clarify this usage of the term "freedom."

Licensing Terminology. So far we have referred to ours and other relevant software such as Linux[6] as free software, however, this needs to be unpacked, as there are multiple conflations which we didn't have the space to previously address.

The English word "free" is polysemous. One adjectival sense of "free" is 'at no monetary cost' (gratis). Another such sense is 'with little or no restriction' (libre)(22) There are at least two prominent types of software that are often free in one or both of these senses:

- **Open Source Software (OSS)**
  - Usually libre (free as in "freedom"[7])
  - Often (but not necessarily) gratis (free as in beer)
  - Emphasizes making good and useful software

- **Free Software (Free/Libre)**
  - Always libre (free as in "freedom"[7])
  - Often (but not necessarily) gratis (free as in beer)
  - Emphasizes a maximalist approach to "freedom" of the end user
  - A proper subset of open source software

Where we have used previously in this paper the term "free software," what we meant is in fact closer to: "free and open source software" (FOSS) a.k.a. "free/libre open source software" (FLOSS). The umbrella term free/libre open source software (FLOSS) is our preferred term for the whole.

There is also an additional optional principle sometimes found in both OSS and free libre software licenses known as copyleft.

> "Copyleft is the legal technique of granting certain freedoms over copies of copyrighted works with the requirement that the same rights be preserved in derivative works."(42)

The FRDCSA and FLP are licensed as free libre software, under the terms of the GNU General Public License (GPL), which makes use of the copyleft principle.[8]

[6] Linux is usually referred to as GNU/Linux in free software circles

[7] Using the Free Software Foundation's usage of the term here

[8] However, FLP and FRDCSA both make use of an extensive number of external open source software systems

By releasing GPL we ensure that it is libre, gratis and copyleft. By releasing libre (or free as in "freedom") we wish to ensure that end users are able to make the modifications they want, for two reasons. First, so that its capabilities belong squarely to the community at large.[9] Second, so FLP may better adapt to local circumstances. By releasing gratis (or free as in beer) we intend to remove barriers to entry, democratize access, and to ensure the widest possible distribution. By releasing copyleft, we intend for the capabilities of the system to always remain with the community. But our usage of copyleft also is partially pragmatic, because it helps the system to snowball in size.[10] Because all modifications must be released under the same terms, it dovetails with the previously mentioned 'bigger is better' heuristic for the creation of more powerful software systems. Thus, by releasing our software libre, gratis and copyleft, we ensure that the software itself and all derivative works are released under the same terms, creating an acceleratingly powerful system for *all* end users.

In conclusion, because software development is parallelizable and software is non-rivalrous / ZMC, libre, and optionally free and copyleft, it is feasible to more easily create and frictionlessly redistribute enormous AI software systems that entail solutions that in some way begin to address societal needs at scale. Free/libre copyleft licenses such as the GPL are thus in a certain sense a practical solution to artificial intelligence[11] and simultaneously a means to employ it in service of society.

Technical Motivation

From a technical perspective, in order to develop FLP we are looking necessarily at modeling reality symbolically and doing planning, theorem proving, formal verification and other types of reasoning on that (blackboard) representation.

Originally, we approached the reasoning subproblem using automated theorem proving (ATP) systems, several(41) (10) of which have been integrated into FRDCSA. However, this approach had several drawbacks, including performance issues and usually an unnecessary restriction to purely logical / declarative reasoning (which omitted the benefits of extra-logical / operational / procedural reasoning).

Due to the 'bigger is better' heuristic, we were drawn naturally initially to the well-known Cyc system, which provides extra-logical reasoning, ATP capabilities, and an enormous commonsense ontology and corresponding axioms with which to model and reason with the real world. However, Cyc itself is proprietary,[12] which is fundamentally incompatible with the GPL license. So we began to reimplement Cyc into a free/libre replacement. Our reimplementation was not a complete success, but yielded many tools which allow Cyc to integrate with FRDCSA.

However, there is another effort containing a free/libre software partial reimplementation of Cyc which is much more advanced. In 23 years of collecting FLOSS artificial intelligence systems, we have found no other project comparable to the Logicmoo AGI(26) (23) system under development by Douglas Miles. Logicmoo has pioneered many critical techniques, including what we term "the PrologCyc discipline," which involves the use of terminology, methodologies, etc. in ways consistent with the Cyc project, but within Prolog.(36)

[9] Community ownership (versus proprietary) would probably enhance buy-in

[10] Ever increasing hardware capabilities also support this snowball effect

[11] Richard Stallman (who wrote the widely-used GPL license, and previously worked at the MIT Artificial Intelligence Laboratory) appeared unaware of his contribution in this respect to AI.

[12] There is freer version of Cyc called OpenCyc, of which we do however make use

Ten years ago, we made the transition to the PrologCyc discipline. Progress on FLP then began in earnest. Fortunately, one feature that Logicmoo and the PrologCyc discipline provides is a comprehensive semantic world simulation. Logicmoo is continuously improving, and hence does not have a completely stable API at present, so we have partially reimplemented aspects of Logicmoo in FLP in the interim. However, both currently can communicate using remote procedure calls.

Design

FLP's Main Planning Systems of Systems

We want to discuss at a higher level of abstraction the present and future capabilities of FLP without weighing down the discussion by referring to all the different systems and technologies in use.[15][16]

In terms of planning technology, the main desired capability is to sync world state in order to plan over it. World state could be thought of roughly as the set of all facts obtaining in a given real-life situation or context. Keeping the digital twin up-to-date by syncing the real world to the digital twin is paramount.

Once synced, various planning programs operate on the world state. Planners generate temporal plans to be executed. The plans are then validated and handed off to an executor which is responsible for walking the user through the plan. At all times the system tries to ensure that plan step preconditions are met by the real world and plan step effects have occurred in the real world.

Future work involves precise monitoring of the conditions under which plans cease to work. If the planning domain is then still valid, it attempts to resync the digital twin with the real world, and then initiate replanning.

The following feature lists are all nonexhaustive. Almost all such features are accessible from the Web UI. An outdated page (with screenshots) describing FLP and its WebUI is available,[21] as well as a paper on an earlier FLP system, SPSE2.[13]

Financial Planner

The financial planner systems generally are aimed initially towards resolving cash flow problems; however, there is plenty of room for growth. The employed planners are very fast, and when combined eventually with meta-planning, will enable contingency planning for doing what-if scenarios over unexpected financial events.

Completed Features.

- Planning and Reasoning Capabilities
  - Integration of action costs and earning information into generalized action planning
  - Full temporal metric planning domain/problem sets for financial prediction and reasoning
  - The GUI for interactive plan execution
    - Checking simple preconditions
    - Propagating simple effects
      - Extraction of world state deltas for plan steps
  - Viewing and editing temporal world state and domain/problem
- Calendaring and Recurrences

  - A webpage showing:
    - Predicted transactions' date, description, debit or credit and running balance
      - In list form
      - Graphs of running balances vs time
  - Calendar displaying expected transactions
  - The integration of financial alerts with various event calendars
  - Text-to-speech based alarms for financial (and other) recurrences
  - Programmable (constraint logic programming) calendrical recurrences, supporting dow/1, onDate/1, through/1, etc.
  - Financial recurrences, supporting, can, neg(onTime), months, date-ranges
  - Detection of recurrences and prediction of future events from logs
- Financial Records
  - Code for extrapolating auto-debits from banks' financial record exports
  - Reconciling bank record exports with reported purchases
- Order Tracking
  - Tracking state of deliveries

In Progress Features.

- Interfaces
  - The WebUI for interactive plan execution
  - A webpage showing:
    - Actual account balances
    - Actual account history
  - A page for editing financial recurrences
  - Debt manager
  - Bill payment system
  - Receipt tracker
    - Online
    - Brick and mortar
  - Integration with inventory and pantry management
  - A resource-manager for ensuring stocking / reordering of all supplies
- Actions
  - Automatically check bank account balances
  - Adding alerts for pending overdrafts

Planned Features.

- Cash flow
  - Computing a monthly budget
  - Creating goals as needed to raise enough money to reach the safety buffer
- Planning
  - Checking and propagating complex preconditions and effects
  - "Meta-planning" (planning to plan) to develop contingent plans, etc.
- Retrospective analysis of correctness of previous predictions
  - To detect missing or incorrect information in the domain
- An argumentation-based purchase decision support system

  - Promoting ethical consumption
  - Reasoning about users' needs
  - Whether potential purchase is necessary and in budget, and if so:
    * which product or supplier to use
    * payment plans
    * etc.

Meal Planner

The meal planner's[28] chief goals are to improve nutrition and taste, and decrease cost and effort. It currently assumes food products are from its databases concerning only the United States. It also at present mostly is limited to ordering and reordering from a particular major grocery retailer, and is lacking generalized food acquisition planning.

Domains for the meal planner[6][7] will be further integrated with domains for transportation,[13] building hours, and other APIs, in order to enable future more difficult food provision scenarios cases, such as are involved with homelessness, lack of transportation, absolute poverty, etc.

There are a few different subsystems which make up the locus of meal planning features. We mention the gourmet-formalog-standalone[1] system in particular, which does not yet generate plans like some of our others, but which is the newest, best and soon to be most powerful planner. Moreover, it is capable of running on its own in Docker, and has been released separately for convenience.

Completed Features.

- Information management
  - A WebUI for scanning products with a barcode scanner
  - Extraction of product information for barcoded products
    * Nutrition
    * Serving size
  - Loading of all databases in a few seconds
    * Two enormous databases for products and nutrition
    * A recipe database with 150,000 recipes
- Planning and execution
  - Full temporal metric domain for generating meal plans based on nutritional requirement profiles
  - Multiple meal planners
  - An interactive cooking assistant which walks through cooking plan steps
- Fridge/freezer temperature and door sensors
  - Tracking pantry inventory changes
  - Ensuring food safety
  - Help regulating food access and thus portion control
    * Ties into a scoring/gamification component

In Progress Features.

- Support for specialty diets (for example low-FODMAP and GERD)

[13] The first FRDCSA planning domain integration experiment involved tying together our bus planner with grocery shopping

- A food ontology
  - Knowledge of food substitutions (including cancellation, addition, and changes)
  - Knowledge of properties of food
    - Storage
    - Allergies
    - Food similarity through vector representations
  - Mapping
    - Recipe ingredients to ontology entries
      - Normalization of ingredients (including errors, such as skinnless -> skinless)
    - Nutrition info
    - Barcode information
  - For use with recipe recommendation system
  - General integration with OpenCyc
  - Integration of a food ingredient toxicity database
- Parsers
  - A parser for parsing natural language recipes into planning problems
  - Several different data normalizing parsers of each of the following kind:
    - Ingredients to food
    - Food data to foods
    - Branded foods to ingredients
- Recipe manager
- Recommender system / collaborative filtering over recipes, ingredients, etc.
- A planning tool to plan to avoid food spoilage and expiration in various storage conditions
  - Fridge/freezer sensor activity
    - Ask the user why they triggered the fridge or freezer sensors
    - Track how long food is likely to stay good based on how long it has been in different conditions

Planned Features.

- Integration of shopping, prep, cooking, cleaning into larger life plan
- Meal planning
  - Specialty diets (medical or ethical)
    - Help with being a vegetarian, vegan, pescatarian, etc.
    - Help with most medical diets (diabetic, missing gall bladder, etc.)
  - Freezer cooking
    - Reduces by 4X both cost and prep time
  - Interactive planning, scheduling and execution of cooking process through semi-auto translating recipes into planning domain/problem
  - Automatic learning of food preferences, detection of pickiness, etc.
  - Determine when the user is going to get tired of something
    - Avoid getting tired of recipes, ingredients, cuisines, etc.
    - Plan to avoid food products that have the same kinds of ingredients
  - Generating comprehensive meal plans with minimal user effort
    - Able to generate multiple options
- Pantry inventory management

  - Help with physical space organization
  - Ability to source ingredients according to user's preference
    - Local, organic, imported, etc.
  - Automatic shopping list / order generation (automatic reordering)
  - Automatic detection of inventory items which have been recalled or linked to and disease outbreaks, based on upstream crypto-signed declarations
  - Proper handling of food items lacking barcodes
- Recipe management
  - Adding an additional repository of another 150,000 recipes
  - Full integration with recipes
    - Eliminating limitations due to temporary working assumptions
      - For instance, 1 serving per meal
  - Able to rate recipes and food preferences
- User modeling
  - Self-discipline coach
    - Understanding a user's psychological relationship to food and food style/preferences
  - Macros planning and counting
  - Checking on symptoms of conditions known to affect user after eating, such as acid reflux, lactose intolerance, etc., to figure out which food sensitivities the user may have in practice
  - Keep detailed nutritional records
    - Ensure privacy of user's diet information
    - Cross-reference with health logs
      - Creation of a body of empirical scientific nutrition knowledge
  - Incorporation of results of genetic testing for which diet is best for the user's body
- Prepping
  - Stock enough provisions to survive n-months
  - Rotating emergency food and water provisions
    - Consuming the oldest staples first to prevent victuals from expiring
- Miscellaneous
  - Food pantry patronage
  - A grocery bill splitter

Health Planner

The last representative system of systems is the health planner. Although there are a few well-developed subsystems, the health planner is still in a relatively early state of development.

Completed Features.

- Appointments
  - Displaying upcoming appointments on a calendar
  - Verbal reminders
- Documentation
  - A document management system / electronic filing system for medical records
- Alexa speech interface

  - Recording medical symptoms
  - Noting when medication and supplements are taken, exercise is done, etc.
  - Q&A system - e.g. "When is the next doctor's visit?"
- Mental health
  - Psychometric reporting
  - Habit tracking, and installation/removal
  - Low productivity amelioration
    - Matching symptoms (using textual entailment recognition) with corresponding counter-strategies
  - Recording and reframing negative self talk with large language models (LLMs)

In Progress Features.

- Medical adherence
  - Tracking of doctors' orders and other medical records in our document management system
  - Formalization of these documents (into PrologCyc)
- Maintaining situational awareness of condition development
  - Administering treatments that are:
    - Preventative
    - Ongoing
    - Restorative
- Integration of planning for medication pharmacokinetics[(3)]

Planned Features.

- An expert system for diagnosing medical problems
  - Bring otherwise unavailable medical knowledge to bear in lieu of a physician
    - Present to and reconcile with doctors' opinions (argumentation-based)
- Diagnostic system immediately wraps into an execution component
  - Large set of standard operating procedures (SOP) and courses of action (COA)
    - E.g. Instruction for CPR using our intelligent tutoring systems
      - Just in time / as needed
      - Ahead of time
  - Integration with telemedicine / remote doctor systems
- Effective and timely dissemination of medical "rules" and knowledge
  - Integration with other telemedicine and remote doctors systems
  - AI-based medical fact checkers
  - A database of images and computer vision tools for diagnostic and explanatory assistance
- Planning, scheduling and execution
  - Medical adherence assistant (for following doctor's orders) through the document management system, machine reading comprehension (MRC), question answering (QA), and conversion to domains/problems for input to the planning system
  - Appointments
    - Ensure coordination, and not overloading the schedule
  - Reminders for all avenues of professional medical health, and classifier for which to

use, for instance, ER, urgent care, phone on-call nurse/doctor, etc.
  - Integration with financial planner and inventory manager to manage health product supplies
  - Hospital packing list, helping one to bring everything they need on an emergency visit to the hospital
- User modeling
  - Creation of an inventory of all medical conditions affecting a user
  - List conditions, meds, and their known effects
    - Inventory their meds and what they do, good and bad, in the system
    - Inventory their providers (who they are, what they do, advise, frequency of visits)
    - Make a note of when someone starts medications and make sure that they are always taking stock of how they feel
  - Symptom tracker
    - Monitor for possible symptoms whenever taking new medications
    - Relevant symptoms automatically get pulled into the report to tell the doctor at the next doctor's visit
      - Places into the dialog temporal plan to tell/ask doctor, walks user through via cell phone
      - Records doctors' answers
    - A "simultaneous localization and mapping" (SLAM) algorithm for situational awareness of disease present and estimated future medical state[14]
    - Help with prevention of conditions with known risk factors, to take measures that are not bad in and of themselves that that would help nevertheless with said condition
  - Keeping proper personal health records, such as tracking: nutrition, exercise, medications, supplements, doctor's visits, immunizations, hygiene, etc.
  - Trying to know more about the situation, to determine when the user might have a valid health concern that the doctor overlooked or is unaware of
- Medication manager
  - Complete constraint logic programming version of medication manager that knows time windows when we can access medications, including hours and availability of pharmacy and clinic
  - Know when meds will run out, and ensure an appointment with the prescribing doctor, or recommend calling in for a refill
  - Know ahead of time which meds cannot be skipped, or if necessary, how long they may be skipped
  - Add medication reordering to the user's agenda, provide phone numbers, make it so it cannot be removed from agenda, not postponable, etc.
  - Use recurrences to run checks on medications every day
  - Notice when the user forgot to refill a prescription or enter that into the system
- Research
  - Research system for finding research that validates or invalidates current course of treatment

[14] For instance, using propagating conditional densities

  - Evidence-based medicine
    * Empirically determine effectiveness of remedies
    * Share anonymous health information for use in data mining and ML, using anonymization or secure multi-party computation

Other Planners and Supporting Features

We have focused thus far on three representative systems, and have gathered some features that are either completed, in progress, and planned relating to each. Because they have very broad scope, their intended missions could support very large lists of features.

We have mentioned some of the systems which assist with finances, physical and mental health, diet and exercise, doctor's visits and orders, and medications. We have precluded mentioning other interrelated FLP systems, including those assisting with: employment, time management, calendaring, planning, scheduling, execution, organization, inventory management, document management, executive function, transportation, shopping and errands, home and equipment maintenance, self-discipline, managing to-do lists, note-taking, interpersonal relationships, emergency preparedness, and so on.

However, we haven't finished properly categorizing FRDCSA's hundreds of thousands of to-do list entries. There are more features intended to be integrated than are mentioned here. Because the system is based on the versatile, scalable and extensible PrologCYC discipline, it can grow to easily and efficiently support such additional capabilities.

Discussion

Critiques of FLP and Responses

Information Security Issues. Because the system uses a knowledge-based digital twin, it models its users' lives in a monolithic data "vault," currently the Prolog logicbase. The basic idea of FLP is to achieve what could be termed "information fusion." (Here we use the term not just in the narrower sense of integration of diverse information sources, but as analogously to fusion (or more accurately, fission) power - setting off chain reactions of logical inferences over comprehensive digital twins / active KBs to achieve near-total situational awareness).

But this "information fusion" is simultaneously one of FLP's most powerful features and also presently one of its biggest weaknesses.

A chief critique of FLP is that there simply remains too much personal information stored in that vault. If it ever were compromised, every security it provides its users would collapse, seriously threatening the well-being of its users.

These privacy and security weaknesses must be resolved before FLP is usable. Below, we discuss several appropriate remedies or mitigations to this central information security problem which preserve the utility of FLP, and can make good on its goal of concluding the current "information dark age:"

- Adhering to HIPAA and other relevant information security standards
- Putting controls on and screening of what kind of personal information may enter the system
- Sandboxing the Prolog factbase, and limiting remote Prolog agent access
- Segmenting the factbase into "contexts" akin to Cyc microtheories

- Air-gapping[15]
  - Interactively walking users through the process of encrypting and air-gapping their vault
  - Distributing vault system upgrades to air-gapped machines through blu-ray discs, or other one-way data-transmission
  - Providing adversarial planners and other cyber-physical IDSes (intrusion detection systems) and IPSes (intrusion prevention systems)
- Other access control measures to ensure the confidentiality, integrity and availability of that information(30)
  - Using encryption on data that needs to be available online remotely:
    - Data-at-rest encryption
    - In-memory encryption
    - Full-disk encryption
  - Planning to use various security methodology with the software:
    - Principle of least privilege
    - Additional security partitioning

"Death by GPS". The most apt metaphor for the FLP is a "GPS for one's life." Unfortunately, as with GPSes, there could be a tendency to fall into one of several errors related to becoming over-dependent upon and uncritically relying on information from the system.(11)

- FLP should be able to test that they are not merely doing as instructed, either by withholding instruction, or giving them small but unnecessary tasks, etc.
- FLP should wear an intelligent tutoring system hat, and train and test its users on critical thinking skills, independent thought and so forth
- If FLP is sufficient generally to protect the user, then it necessarily is sufficient to explicitly train the user generally to avoid the problems associated with "Death by GPS"

Corruption of Facts and Rules. A very serious problem is that of either unintentional corruption of, or deliberate tampering / "poisoning" of, the rulebases, similarly to a recent news story which involved the Alexa system accidentally daring a child to perform a dangerous challenge.

- Similar to "Death By GPS," some critical thinking skills (and perhaps age gates) ought to be established first.
- Trust management mechanisms, and traditional cybersecurity:
  - It is desirable that most or all rules be cryptographically signed declarations by responsible parties
  - The ability to say which sources, concepts etc. one trusts
  - The use of a truth maintenance system (TMS) with the ability to deduce whether subsequent rules are consistent with prior beliefs, and to employ belief revision when appropriate.

Criminal Misuse. Knowledge of the good is almost always invertible. For instance, if one knows the conditions of life, then it is possible to plan to preclude them. Moreover, if such a system FLP is at all powerful, then that power may be misapplied.

- FLP intends to develop "livingry"(25) as opposed to weaponry, to repurpose dual-use software for use in a social safety net. It aims to produce products which are empowering but have low

[15] Mostly completed

lethality.
- Good tends to be constructive, and bad, destructive, and hence bad may self-destruct.[16]
- A proliferation of "intelligence" would tend to rule out unsound or immoral options, because wanton harm is a miscalculation.
- The active life planning / intelligence features may be federated to provide a kind of collective peer-to-peer, checks-and-balances form of security.
- The system contains reasoners or ethical governers for deontic logic, and attempts to let people know when their value systems are being violated.

Legal Liability. The issue of legal liability is superficially handled by the license terms 16 (Limitation of Liability) of the GNU General Public License Version 3, which disclaims any liability for the use of the software. However, the larger issue of potential for harm due to the software remains.
- As with "criminal misuse," checks and balances would help to restrain unsound or immoral actions.
- Bringing the code up to standard regarding standards such as HIPAA
- The rest of this discussion is beyond scope of this paper.

The Digital Divide renders FLP Futile. The next logical critique of FLP is access. Some would argue that the digital divide renders FLP futile, that we cannot help everyone:
- We can nevertheless help some
- The Digital Divide is a red herring
  - Recycling: relatively powerful computers (able to be thin (web) clients or more) are constantly being literally thrown out
  - Commodity computation: a Raspberry Pi variant and MicroSD Card can be bought for ~$15 USD
  - Cell phones now are nearly ubiquitous
    - Cellphones for the homeless are life lines:
      - One study in the US showed 70% of a local homeless population had cell phones
      - Jobs: cell phones allow people to apply for work
      - Transport: allow users to plan for transportation
      - Other uses: millions of other use-cases
      - Cell phones would enable access to FLP
  - People who lack direct access to computers can be helped by intermediaries who do have access, such as government workers

"No One Size Fits All". It may be objected that no one size fits all, and that hence a life planner is impossible because there is no single program which can handle all use-cases. However:
- Many problems are actually quite routine
- Programs can use conditional logic and other algorithms which are responsive to differences
- FRDCSA is actually a sequence of systems (each improving on the previous)
- FRDCSA is libre software that can be modified to purpose

[16] Paraphrasing a commment by Jachen Duschletta, pers. comm.

Monoculture. It is well known that diversity adds evolutionary robustness to a population, and that rare alleles contain the bulk of a population's genetic information. Thus, the critique is that, should a single universal life planning system become standard, flaws in that system could be magnified and potentially catastrophic.

- There is no reason to suppose that FLP will be the only such system out there.
- The ability of the FLP (and all FLOSS software) to fractionalize into separate lineages, due to forking of codebases, works in its favor in this case.
  - We intend to make the entire project (including DevOps for the project itself) completely forkable. Work is underway on a self-replicating installer.[19]

Software Maintenance. With a large system like FRDCSA, software maintenance becomes a valid issue. What is, for instance, the "bus-factor" for core FLOSS systems that FLP and FRDCSA rely on?

- Recent efforts to address these problems have been begun by DARPA.[24]
- A useful free software program is likely to have many maintainers.
- Forkable DevOps eases the ability of the system to survive its developers and maintainers

Real-World Effectiveness. Another objection which sometimes arises is whether software, being itself partially intangible, can have any effect on the real world.

- Computer systems control all kinds of things in the real world
  - E.g. access to the monetary system via ATMs
  - Operate at incredible speeds and across vast distances
  - Digital twins are a highly effective means of representing and interacting with the real world in a controlled manner
  - Guide and shape user behavior through apps like FLP

Future Work

Release of FLP 1.0

An earlier, reduced and redacted version of FRDCSA and FLP already has been released in 2020, called Panoply.[20][2] However, this release predates the completion of the bulk of FLP's features, It is not a simple matter to backport those features from current private FRDCSA/FLP to Panoply.

The chief obstacle to progress is difficulty completely releasing a live and current branch of the FRDCSA/FLP system. There is not a clear logic / data distinction in the system yet, and we have been dogfooding[14] the application. Consequently, there are private data strewn throughout the system. For the past couple years we have been working on a set of integrated tools for releasing the entire codebase. Based on our earlier experience releasing Panoply, we tentatively have set a two-year time frame for completion of these tools and the next release.

Development of FLP 2.0

The utility of the FLP 1.0 system is constrained by lack of proper security measures. While we continue to work towards the FLP 1.0 release, it is clear that FLP needs a thorough refactoring and possibly rewrite. It should be possible to take large swaths of code from FLP, bring them under

test, refactor them into appropriate structures such as Prolog modules, make proper documentation, and verify adherence with all applicable standards.

Given the significant motivation behind the project, we will continue implementing regardless of any assistance received. However, as the author is the copyright holder, it is possible that we could relicense the relevant codebases under a dual permissive[17]/GPL license arrangement. Then, we could commercialize the project in order to help fund its development and reach potential users quicker, while retaining a free/libre version which would achieve the goals of community ownership and buy-in, software adaptation, low barrier to entry, democratized access, and widest possible distribution.

Conclusion

We have discussed how an alignment of beneficial factors (the dovetailing of software non-rivalry, free software licensing and development, and the 'bigger is better' heuristic for AI) makes possible the creation of a universal and ubiquitous free software social safety net. Consequently, this motivated much work over the last quarter-century on FRDCSA, and over the last decade on FLP itself. This latter system is intended to help manage the logistics of daily living, including for domains such as financial, nutrition and health. We have created a digital twin system / blackboard system (using the capable PrologCyc discipline), and built many useful services. Much initial progress towards creating an effective system for resolving life-critical issues has been accomplished.

We have looked also at some possible objections to the project, as well as some associated rebuttals. Some chief difficulties for the project as it stands include information security and difficulty properly releasing the software. Although we fully expect to overcome these difficulties, various kinds of collaboration and/or assistance could speed up the time frame. This potentially could have safety-critical effects, and hopefully could assist efforts with global mitigation of some wicked problems including poverty, hunger, and disease.

References

aindilis/gourmet-formalog-standalone: A fork of gourmet-formalog designed to be independent of frdcsa, and accessed through pengines or repl. (n.d.). Retrieved 2022-12-01, from `https://github.com/aindilis/gourmet-formalog-standalone`

aindilis - vagrant cloud. (n.d.). Retrieved 2022-12-01, from `https://app.vagrantup.com/aindilis`

Alaboud, F. K., & Coles, A. (2021, May). Personalized medication and activity planning in pddl+. *Proceedings of the International Conference on Automated Planning and Scheduling*, *29*(1), 492-500. Retrieved from `https://ojs.aaai.org/index.php/ICAPS/article/view/3514` doi: 10.1609/icaps.v29i1.3514

"automation for debian packaging" - jelmer vernooij (lca 2022 online) - youtube. (n.d.). Retrieved 2022-12-01, from `https://www.youtube.com/watch?v=qs6zGDZNvZw`

Benefits of making packages. (n.d.). Retrieved 2022-12-01, from `https://frdcsa.org/~andrewdo/writings/benefits-of-making-packages.html`

[17] Or even proprietary

caloriesinge.d.verb. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org/~andrewdo/projects/mealplanning/caloriesingle.d.verb.txt

caloriesinge.p.verb. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org/~andrewdo/projects/mealplanning/caloriesingle.p.verb.txt

Chaitin, G. J. (1974, jul). Information-theoretic limitations of formal systems. *J. ACM*, *21*(3), 403–424. Retrieved from https://doi.org/10.1145/321832.321839 doi: 10.1145/321832.321839

Corrigan, R. (2007, September). Colmcille and the battle of the book: Technology, law and access to knowledge in 6th century ireland. In *Gikii 2 workshop on the intersections between law, technology and popular culture at university college london, september 19th, 2007.* Retrieved from http://oro.open.ac.uk/10332/ (GikII is annual closed workshop chaired by Professor Lilian Edwards. By invitation only, the workshop focusses on law in popular culture and governance in an online world.)

Cyc | the next generation of enterprise ai. (n.d.). Retrieved 2022-12-01, from https://cyc.com/

Death by gps - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Death_by_GPS

Digital twin - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Digital_twin

Dougherty, A. J. (2011). Temporal planning and inferencing for personal task management with spse2. Retrieved from https://frdcsa.org/visual-aid/pdf/Temporal-Planning-and-Inferencing-for-Personal-Task-Management-with-SPSE2.pdf

Eating your own dog food - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Eating_your_own_dog_food

flp/referencemanual.md at main aindilis/flp github. (n.d.-a). Retrieved 2022-12-01, from https://github.com/aindilis/flp/blob/main/ReferenceManual.md

flp/referencemanual.md at main aindilis/flp github. (n.d.-b). Retrieved 2022-12-01, from https://github.com/aindilis/flp/blob/main/ReferenceManual.md#major-technologies-used

Frdcsa: Packager. (n.d.). Retrieved 2022-12-01, from https://altruisticsoftware.org/frdcsa/internal/packager/

Frdcsa project homepage. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org

Github - aindilis/frdcsa-installer: The install scripts for frdcsa (panoply). (n.d.). Retrieved 2022-12-01, from https://github.com/aindilis/frdcsa-installer

Github - aindilis/frdcsa-panoply-git-20200329: How to run panoply git gnu/linux (the version of frdcsa made by combining all of the redacted github frdcsa codebases). (n.d.). Retrieved 2022-12-01, from https://github.com/aindilis/frdcsa-panoply-git-20200329

Github - aindilis/free-life-planner: Free life planner: An ai tool for helping with planning for day-to-day life. (n.d.). Retrieved 2022-12-01, from https://github.com/aindilis/free-life-planner

Gratis versus libre - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Gratis_versus_libre

Help us create agi - developer - xwiki. (n.d.). Retrieved 2022-12-01, from https://logicmoo.org/xwiki/bin/view/Main/Developer/

Hybrid ai to protect integrity of open source code (socialcyber). (n.d.). Retrieved 2022-12-01, from https://www.darpa.mil/program/hybrid-ai-to-protect-integrity-of-open-source-code

Livingry (buckminster fuller institute). (n.d.). Retrieved 2022-12-01, from https://www.bfi.org/about-fuller/big-ideas/livingry/

Logicmoo artificial human intelligence. (n.d.). Retrieved 2022-12-01, from https://logicmoo.org/public/

Lti mourns loss of founder and director jaime carbonell | carnegie mellon university - language technologies institute. (n.d.). Retrieved 2022-12-01, from https://www.lti.cs.cmu.edu/news/lti-mourns-loss-founder-and-director-jaime-carbonell

Meal planning resources. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org/~andrewdo/WebWiki/MealPlanningResources.html

Ognibuild. (n.d.). Retrieved 2022-12-01, from https://www.jelmer.uk/ognibuild.html

Pioneer october contest - flp update - week 3. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org/~andrewdo/writings/flp-update-4.html

Practical reason (stanford encyclopedia of philosophy). (n.d.). Retrieved 2022-12-01, from https://plato.stanford.edu/entries/practical-reason/

Rivalry (economics). (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Rivalry_(economics)#Anti-rivalry

Rivalry (economics) - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Rivalry_(economics)

Statement of solution concept. (n.d.). Retrieved 2022-12-01, from https://frdcsa.org/~andrewdo/presentation-6.jpg

sto0pkid/godelian: Godelian stuff in agda. (n.d.). Retrieved 2022-12-01, from https://github.com/sto0pkid/Godelian

Swi-prolog. (n.d.). Retrieved 2022-12-01, from https://www.swi-prolog.org/

Tecuci, G., Boicu, M., Marcu, D., Bowman, M., Ciucu, F., & Levcovici, C. (2000). Rapid development of a high performance knowledge base for course of action critiquing. In *Aaai/iaai.*

Thousands of debian packages updated from their upstream git repository. (n.d.). Retrieved 2022-12-01, from https://www.jelmer.uk/fresh-builds.html

tool lists / verification synthesis.md at main johnyf/tool lists github. (n.d.). Retrieved 2022-12-01, from https://github.com/johnyf/tool_lists/blob/main/verification_synthesis.md

The upstream ontologist. (n.d.). Retrieved 2022-12-01, from https://www.jelmer.uk/upstream-ontologist.html

Vampire. (n.d.). Retrieved 2022-12-01, from https://vprover.github.io/

What is copyleft? - gnu project - free software foundation. (n.d.). Retrieved 2022-12-01, from https://www.gnu.org/licenses/copyleft.en.html

Wicked problem - wikipedia. (n.d.). Retrieved 2022-12-01, from https://en.wikipedia.org/wiki/Wicked_problem