Initial Draft Language for Grant Proposal for the Development of HARK 2.0

Purpose: This document contains preliminary language that could be included in a request for funding for the development of "HARK 2.0", a software package for representing, solving, simulating, and estimating dynamic models with heterogeneous agents. It is not formatted as a grant proposal, but instead merely includes fragments of text that can be further developed and refined. The name "HARK 2.0" is used only as a shorthand in this paragraph.

Synopsis: In 2017, the newly founded Econ-ARK project received a generous grant from the Alfred P. Sloan Foundation to develop the `HARK` toolkit, a software package for solving, simulating, and estimating heterogeneous agents macroeconomics models. Using this funding, the structure and presentation of the `HARK` package was professionalized, and a wide variety of canonical models and applications were developed as exemplars for economists to use when developing their own research projects using `HARK`. In the years since, and using NumFocus as our fiscal sponsor, we have received additional funding from other sources (including the Think Forward Institute and T. Rowe Price, as a "no strings attached" corporate sponsor) to continue the work and expand the range of models offered in the `HARK` package, allowing us to achieve our original set of goals.

Over the past seven years, the combination of our experience producing `HARK`, feedback we have received since its inception, and relatively recent developments in other software packages have led the Econ-ARK team to conclude that a new software package and modeling schema is needed to further advance the field and fully realize the vision of a common platform for dynamic structural modelers. We seek funding to create and deploy this new system, which we believe will fill a critical gap in the toolset available to structural economists, thus accelerating the development of models on the frontier of economic research, improving the verifiability of numeric output from such models, and improving communication and collaboration among researchers. In addition to academic researchers, such a platform would be of significant use both to governments (including central banks) in conducting prospective analyses of potential policy actions, and to private financial institutions who wish to make decisions or provide advice that is informed by a rigorous structural model.

Background: Over the past thirty years, economists have increasingly developed dynamic models of the decisions of individual agents (i.e. households or firms) that are intended to be interpreted *quantitatively* rather than only *qualitatively*. Instead of merely characterizing the general properties of the outcomes predicted by a particular theoretical model,

researchers instead seek to quantitatively evaluate its performance in matching past observed outcomes, and to generate quantitatively plausible predictions of the effects of hypothetical future events and policies. This approach to economics is often referred to as structural modeling. When applied to macroeconomic topics, structural modeling involves generating aggregate outcomes as the accumulation of the individual decisions of many agents, each of whom may face different circumstances idiosyncratic to themselves. Hence this subfield is known as heterogeneous agents macroeconomics ("HA macro"), in contrast to the more traditional representative agent ("RA") approach in which idiosyncratic heterogeneity among households and firms is assumed away.

Modern models in both HA macroeconomics and structural microeconomics include stochastic processes that realistically reflect the large uncertainties inherent to the agents' problem– potentially large shocks to their productivity, demand for their labor (or output), health, etc. In combination with risk averse preferences or constraints on access to capital, these sources of uncertainty generate model solutions for optimal behavior that are highly non-linear and require global (rather than local) methods for solving the model. Such methods are computationally complex and costly, requiring careful application of fundamental numeric tools like quadrature methods, functional approximation, and solving systems of non-linear equations. These tools are not usually taught as part of graduate economics training, and it takes years of experience to understand how they should be applied, and how to interpret (and correct) their apparent failures and shortcomings.

While economists have developed a standard and well-adopted set of tools for representing, solving, and examining RA macro models (including especially the Dynare software package), and there is commercially available software for statistical and econometric analysis (e.g. Stata), there is no widely adopted software nor standard for HA models, nor dynamic structural models in general. Each research team instead develops its own bespoke software for each new project, often inheriting or adapting pieces from their own prior work. The code for each model is designed to meet the specific needs of that project, and there is no interoperability or standards for the usage or application of numeric methods. Individual toolkits have been developed for classes of problems and types of analysis, but none is universally applicable and each has limitations on the type of models that can be represented.

Proposal Summary: Econ-ARK seeks funding to develop a language for expressing dynamic structural models (with a particular focus on heterogeneous agents macroeconomics), specifying numeric methods, and describing simulation procedures to generate model out-

put. The language is intended to provide a common format for describing dynamic structural models that will be widely adopted for precisely conveying model content in a human- and machine-readable way, but is independent of the software and code to actually solve and implement the model. To that end, the funding sought will also be used to develop a software package that interprets model statements in the new language, generates code for solving and simulating the model (with the provided numeric method choices), and allows the user to interactively develop and build a structural model and examine its solution and output.

Economic models are usually expressed with a combination of mathematical and natural language: most of the content can be concisely conveyed as a series of equations, inequalities, and other mathematical statements, while a few additional details are provided in plain English outside of the formal statement.[1] This representation of the theoretical model is intended to be readily understood by a human reader. Because dynamic structural models almost surely do not have a closed form solution, they can be solved only approximately using numeric methods. While it is increasingly common for researchers to publicly archive their project code, a description of the numeric methods used is usually omitted from the final published paper;[2] this information is (at best) relegated to an online appendix or (at worst) not documented anywhere outside of the code itself. Moreover, unlike the conventions of mathematically expressing a system of statements to compose a theoretical model, there is no straightforward and complete way to convey such numeric methods and details, even if an economist were so inclined. Even worse, there is no formal relationship between the model as expressed *on paper* and the problem as solved *in code*– the academic refereeing system focuses deeply on the economics and relies on trust with respect to the numerics.

Our proposed modeling language seeks to rectify these systemic issues with the workflow of economic research that uses dynamic structural models. We will provide a common platform for precisely representing the mathematical mechanics of dynamic structural models, providing syntax to specify a wide range of model features. The representation of a model in this language will be both human-readable (via a concise syntax and simple structure) and machine-readable (due to the precision and completeness of the modeling language). If a model specification file is used to generate a numeric solution and model output, a reader or evaluator can be confident that the model presented on paper matches its execution in code.

[1] For example, few models of a person's life-cycle include a precise mathematical representation of what it means for an agent to "die" in the model; the authors trust that the reader understands the implications.

[2] E.g., economists rarely publish information about quadrature methods, the discretization of a continuous state variable, or convergence criteria.

Furthermore, our language will include a format for specifying the methods and choices used to solve the model numerically, transparently providing this information alongside the "pure" mathematical content of the model. In combination with our proposed software package, we seek to provide a platform for testing and evaluating the performance of a numeric solution to a theoretical model.[3]

Extant Tools: Several software packages already exist that are *related to* our proposed modeling framework, but are *insufficient* individually or in combination to fulfill our vision. We will briefly describe each in turn, addressing the capabilities and limitations of the software, as well as how its scope and design relate to the work for which we seek funding.

Most prominently, `Dynare` is a software package for the Matlab programming environment that has been in continuous development and use since 1994, and has been widely adopted for work with traditional representative agent macroeconomic models. In `Dynare`, a user first specifies their model in a file, declaring each variable, parameter, random shock, law of motion, etc. They then pass the model file to `Dynare`, which fully automates the process of solving for model behavior and provides an API for examining and processing model output. The format for a `Dynare` model file is relatively straightforward (and well documented), with an organizational structure that makes it reasonably readable by a human. Because of the package's wide adoption among RA modelers, `Dynare` model files can be (and regularly are) exchanged among different research teams. Over nearly thirty years of updating and improvement, the range of model features that can be handled by `Dynare` has significantly expanded, as has the set of numeric methods that it can employ.

Undoubtedly, `Dynare` is a strong inspiration for our own vision, and a touchstone among macroeconomists. Indeed, when the `HARK` package is described to economists, they often interpret it *through the lens* of `Dynare` and its capabilities for RA models. That is, they initially believe that `HARK` models are specified in model files and solved automatically by the package. Rather, `HARK` is a platform in which the solution method for each microeconomic model must be hand-coded by the user, but provides an environment that makes it relatively easy to introduce arbitrary *ex ante* heterogeneity among agents (as well as represent large populations with *ex post* heterogeneity arising from idiosyncratic shocks), with a wide array of example models for users to adapt for their own purposes.

[3]E.g., a tool for computing the accuracy of an approximate solution as measured by its adherence to a known characteristic of the solution, such as an automatic test of Euler equation errors.

This mistaken interpretation of `HARK` is in fact key evidence in support of the great need for the platform we are proposing: macroeconomists broadly understand the *limitations* of `Dynare` and hope for its extension to the frontier of heterogeneous agents modeling– the new generation of HA models don't use `Dynare` because they *can't*. Simply put, `Dynare` is designed solely around representative agent models, that might have a representative consumer, representative firm, and/or a representative bank, but *never* have any concept of a *multitude* of consumers, firms, or banks, each with their own circumstances. We seek to develop a modeling language and software platform that is akin to "`Dynare` for HA", for which so many of our colleagues have expressed hope. Because of the fundamental differences in the *kind* of problems posed by HA vs RA models (and hence the kind of solution methods and numeric tools involved), `Dynare` cannot simply be extended to encompass HA macroeconomic models. This is evidenced by the fact that despite its ubiquity, familiarity, and long history, no one has attempted to extend the package in this way. Instead, an entirely new system is needed, albeit one inspired by the scope and ongoing extension of `Dynare`.

Over the past decade, Pablo Winant has developed[4] the `dolo` package for the Python programming language, focusing on dynamic stochastic general equilibrium (DSGE) models. Like `Dynare`, models in `dolo` are passed as a model file, declaring state variables, exogenous shocks, control variables, etc, then describing the laws of motion and conditions for optimal controls. After verifying that a valid model has been specified, `dolo` then solves for the optimal policy function automatically and can produce simulated output via its API. More recently, Pablo has extended `dolo` to the `dolark` package (now using the Julia programming language), which incorporates important elements of HA models.

Even as it permits models with non-trivial heterogeneity among agents, the fundamental assumption underlying the structure of `dolark` is that agents have an *infinite horizon* when evaluating future possibilities. That is, agents take the perspective that they will experience an infinite sequence of discrete periods, each of which is structurally identical *but for* the particular circumstances they find themselves in (both idiosyncratically and with respect to aggregate variables) and the realizations of random shocks that period. This modeling approach is sometimes referred to as "perpetual youth", as there is no concept of age or a life-cycle, and is typical in macroeconomic modeling due to its convenient properties and close relationship to RA models. In contrast, our proposed framework goes far beyond infinite horizon models (while still incorporating them) to include decision problems over a finite life-

[4]Pablo is the sole primary developer for `dolo`, while `Dynare`'s development is governed by a wider group of programmers and economists, after being originated by Michel Juillard.

cycle, including those in which the *basic structure* of the problem can change across periods. Moreover, it includes concepts of discrete choice and non-deterministic "flow control", as well as a format for representing the formation of agents' beliefs about endogenous processes.

Even more recently, a team of economists has released the `sequence-jacobian` package for Python, which provides a toolkit for HA macro models using their recently published work on the sequence space Jacobian method ("SSJ", as the toolkit is usually called).[5] A critical complication of HA macro models is that the *state* of the economy at a point in time is an infinite-dimensional object representing the entire distribution of idiosyncratic states of individual agents. That is, future aggregate values depend on the cumulative choices of all individual agents, and the decisions of those agents depend on both their expectation of future aggregate values and their idiosyncratic circumstances; hence a *fully rational* agent needs to know the entire distribution of his fellows' states in order to make a perfectly informed decision. Because this is a clearly infeasible task, economists have focused methods for simplifying the model, usually by (greatly) reducing the dimensionality of the aggregate state ("approximate aggregation"), or by approximating the solution to individuals' microeconomic model with (greatly) reduced dimensionality. In contrast, the SSJ method expresses equilibrium conditions in terms of the *sequence space* of perfect-foresight outcome sequences, rather than in terms of the macroeconomic *state*. Per the authors, this approach allows them to preserve the complexity of microeconomic behavior while quickly and accurately computing the macroeconomic outcomes from perturbations around the steady state ("impulse responses").

The SSJ toolkit provides a framework for representing HA macro models and characterizing the equilibrium conditions as a directed acyclic graph (DAG), specifying model components in "blocks" that flow into each other in a sequence of information. In a canonical HA macro model, one block would be the behavior of individual agents (e.g. households choosing how much to consume vs retain in savings) within the period and another block would be the market-level behavior (e.g. aggregation of individual savings into aggregate capital and determination of factor prices). With sufficient information provided for characterizing the steady state of the model (both in aggregate and individually optimizing policies), the SSJ toolkit automatically (and quickly) computes objects that characterize general equilibrium behavior of the model. These objects in turn can be used to analyze the model's response to any (reasonably small) perturbation around the steady state.

[5]This toolkit is maintained and developed by Bence Bardóczy, Michael Cai, and Matthew Rognlie; Bardóczy and Rognlie are two of the four coauthors on the new paper.

The SSJ method and accompanying toolkit is undoubtedly a massive advance for the field of HA macroeconomics. Both the perspective shift from the *state space* to the *sequence space* and the restructuring of HA models as a DAG are significant conceptual steps, and the provision of software that implements the method will accelerate the development and analysis of HA macro models. However, it is not without its limitations. While the SSJ method can quickly produce *aggregate variable* time series impulse responses to various perturbations, it does not actually produce simulated *microeconomic* data in doing so; the main thrust of the method is to *avoid* doing so. As the authors write in their paper (emphasis added), "Given these [sequence-space] Jacobians, the *underlying heterogeneity no longer matters*: the Jacobians tell us everything that we need to know, to first order, *about the aggregate behavior* of the models heterogeneous agents." General equilibrium behavior derived from the SSJ method thus cannot inform any analysis of microeconomic (e.g. redistributional) consequences of macroeconomic changes. Moreover, while the toolkit automates much of the computation of the steady state and general equilibrium behavior, it does not automatically solve the *microeconomic* problem faced by individual agents (nor does any other extant package, of course).[6] Like `dolo` and `dolark` before it, the SSJ method and toolkit focuses exclusively on infinite horizon microeconomic models, largely forgoing life-cycle considerations (as is typical in most of macroeconomics).

Our proposed modeling framework will build on and advance the structural philosophies of `dynare`, `dolark`, and the SSJ toolkit. Rather than ignore life-cycle models and fundamentally assume an infinite horizon model, our framework is designed to handle life-cycle problems (typical in the structural microeconomics literature) *and* infinite horizon models by explicitly specifying the recursive nature. Inspired by the formalization of a model as a DAG in the SSJ toolkit, models in our propose framework are also specified in "blocks" whose inputs and outputs flow into each other, but more permissive with respect to the structure of the graph. In our vision, extant toolkits for economic modeling can be linked to our software package and used when the specified model (and desired analysis) is appropriate for that package's capabilities.

In our extensive field research prior to beginning work on the new platform, we conducted a thorough search of *other* academic fields, investigating whether a general dynamic modeling

[6] As the developers note in one of their tutorial `jupyter` notebooks for the package, "Backward iteration [of the microeconomic problem] is model specific, and requires careful consideration. [...] As such, we ask users only to provide a recipe for backward iteration[, s]o the main task is to write a **backward step function** that represents the Bellman equation."

schema has already been developed. Despite considerable effort, we found that there is no comparable or related project that could be adapted or expanded for our purposes. Rather, we found that the universe of modeling- and optimization-adjacent software is both diverse and diffuse: a collection of useful but *unconnected* software tools. The lack of a common platform for representing dynamic models and calling tools for handling subsets of them is akin to the lack of cohesion among the various artificial intelligence (AI) and deep learning toolkits that have recently developed. That is, researchers who want to use multiple AI tools must write their own code to link to each one individually, rather than there being any kind of common interface.

In addition to the modeling-oriented software packages, there are general purpose computational packages that are relevant and adjacent to our proposed platform. Most prominently, the `sympy` package for symbolic mathematics in Python has made great progress in recent years. With `sympy`, users can execute algebraic operations on symbolic expressions, including various simplifications and substitutions, solving a system of equations, and automatic differentiation and integration. Critically, it also allows the user to easily transform a symbolic representation of a function into an executable Python function, interfacing with several numeric backends for high performance computation.[7] A key feature of the modeling platform we describe below is that model objects can carry both a (pure) mathematical representation of their content as well as a numeric form (i.e., an approximated digital representation). Hence while `sympy` provides incredible symbolic math capabilities, our proposed package will *apply* these to create a modeling environment for structural economics.

Language Overview: The core element of our proposed modeling language is a "stage", representing a sequence of events for an agent (or actor). An agent's dynamic programming problem is constructed by linking these stages together, potentially recurring on itself. Model stages are meant to compartmentalize information, explicitly specifying the variables that are "inbound" to the stage as information (i.e. "state variables") and objects that can be observed from its successor when solving (e.g. the continuation value function). Each stage has a declaration of its parameters, distributions, functions, and algebraic substitutions. The *values* or *details* of these parameters and distributions do are not required when a block is defined, and can instead be left symbolic.

The dynamic model content of a stage could be as small as one model "step", or it could encompass as much as what economists would ordinarily describe as entire period of the

[7] Options include `mpmath`, `numpy`, `CuPy`, and `jax`, as well as Python's native `math` package.

model. For example, in the context of a basic consumption-saving model with transitory labor income risk and a riskless return, a stage could be as small as going from beginning-of-period capital holdings k_t to bank balances b_t by multiplying the former by return factor R; or it could be the next step, where labor income y_t is drawn from some distribution and added to b_t to yield market resources m_t. Alternatively, a stage could be specified as having *all* of the steps in one period of the consumption-saving problem: earning interest on capital, receiving stochastic labor income, choosing consumption, and retaining non-consumed market resources as assets.

Suppose a user chose to specify the simple consumption-saving model using (say) four very small stages. Under our proposed modeling language, they could further specify that these stages are sequentially linked together. The resulting linked group of stages would itself then represent a stage (the entire period) that could be further linked.

The Econ-ARK team has been developing draft specifications of this modeling language, formatted as YAML files. This allows model information to exist alongside parameterization choices, but not inextricably paired together. Instead, the modeling language will include a directive for assigning parameters (or distribution information, etc) from a YAML dictionary entry to a stage, filling in previously unspecified information. As discussed below, our proposed software package would also allow a user to import a stage into their modeling environment without parameter values, and *then* to assign values.

Our draft language specification includes concepts for flow control among stages, so that (e.g.) an agent's discrete control can govern the *nature* of the next choice they face, or a mortality shock can determine whether they continue to the next period or simply terminate. A group of connected stages thus acts like (and can be represented by) a directional graph among a set of nodes. A graph that loops back on itself is interpreted as an infinite horizon model in which the solution for optimal controls is recursively defined, while a graph that reaches a definite terminal node is a finite lifecycle problem.

The language will also have syntax for how information interacts among agents, including agent-to-agent transfer and market-level aggregation– how idiosyncratic actions and states accumulate into macroeconomic outcomes. In our modeling language, such interactions are conducted by an actor that is also constructed from an assemblage of stages, albeit one that has no *agency*– it makes no choices and has no motivation or preferences. This structure allows for the macroeconomic endogenous objects (e.g. equilibrium prices) to be specified in the same way as microeconomic endogenous objects (e.g. optimal policy functions).

Software Package: The software package we plan to develop will act as an interpreter of the YAML model statements, interactive model development environment, and platform for applications using the models (e.g. structural estimation). Several features of the model specification language will also be included as programming directives in our software package– the user can link stages, create a sequence of periods, assign or change parameters, etc.

Our proposed software package will read in YAML model files and generate representations of model objects using `sympy`'s various capabilities, which now include objects that are simultaneously both a symbolic representation of a function's mathematical form *and* a numeric approximation of it (e.g. a representation of the optimal policy or value function) that can actually be executed. The `sympy` package now has well developed interfaces with various numeric back-ends (including `numpy` and `jax`), so that carrying symbolic representations of the model does not (necessarily) impose grave efficiency costs on computations using them.

Likewise, `sympy`'s automatic differentiation and algebraic substitution capabilities have now been sufficiently well developed that an Econ-ARK team member has prototyped code for generating an algebraic representation of the first order and envelope conditions for an intertemporal optimization problem. These representations can be automatically converted into code that numerically solves the choice problem using the endogenous grid method (EGM). Our software package will thus be able to quickly generate (a draft of) solver code from the model input, which could be used as-is or manually refined by the user; it will also allow the user to specify fully custom solver code.

We have also already prototyped automatic Monte Carlo simulation based on YAML model input, ensuring consistency between intended and actual model behavior. The new version of HARK will support multiple modes of population dynamics, including explicit mortality Monte Carlo, cumulative survival probability weights, and discretizations of continuous state variables (for Markov-style approximations).