DARPA ASKE GE Team M13 Report

Quantifying GE ASKE System (ANSWER + KApEESH) Capability and Performance

Andrew Crapo, Alfredo Gabaldon, Narendra Joshi, Vijay Kumar, Varish Mulwad, Nurali Virani
GE Research
June, 2020

This work is supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990006 and under Agreement No. HR00111990007. This report combines the M13 report for GE TA1 and GE TA2. The latest version of this report is available at https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/blob/master/StatusReports/Phase2/M13/GE_ASKE_M13_Report.pdf

and at

https://github.com/GEGlobalResearch/DARPA-ASKE-TA2/tree/master/Reports/GE_ASKE_M13_Report.pdf

Table of Contents

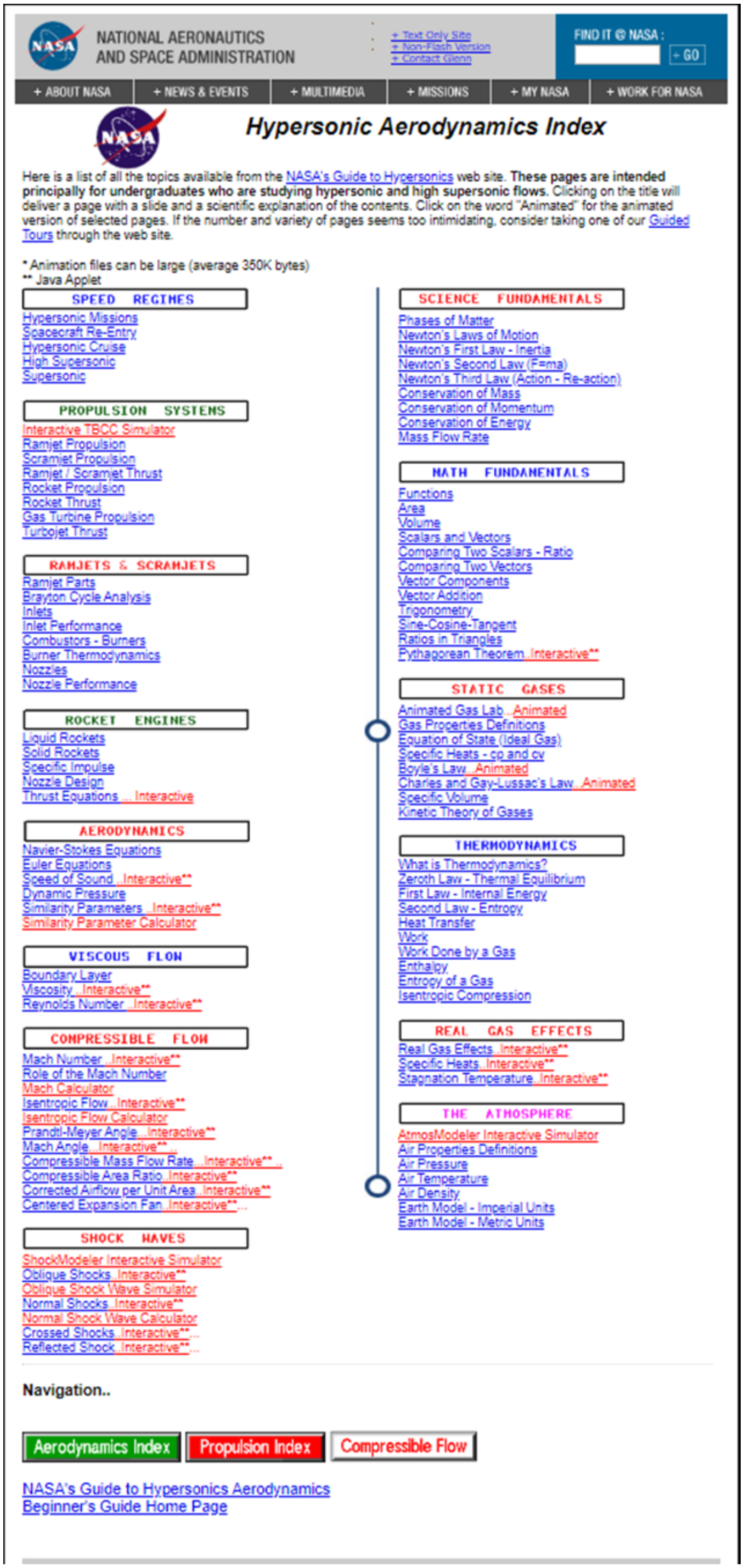
Introduction

The purpose of this report is to quantify the performance of the GE ASKE System (integrating GE systems ANSWER and KApEESH) applied to the target sources of code and text, the NASA Glenn Hypersonics Web Site. We also compare and characterize the computational modelling framework in the GE ASKE system.

Code and Text Sources: NASA Hypersonics Web Site Content

As identified in the GE proposals, our code and text sources come from the NASA Glenn Hypersonics Web Site. The code consists of eight Java applets and the text consists of 63 HTML pages. Figure 1 shows the Hypersonic Aerodynamics Index page, available at https://www.grc.nasa.gov/www/BGH/shorth.html. It has links to the HTML pages listed in Table 1, which have been converted to text and processed as part of our evaluation of performance.

Figure 1: Hypersonic Aerodynamics Index



Hypersonic Aerodynamics Index

Here is a list of all the topics available from the NASA's Guide to Hypersonics web site. **These pages are intended principally for undergraduates who are studying hypersonic and high supersonic flows.** Clicking on the title will deliver a page with a slide and a scientific explanation of the contents. Click on the word "Animated" for the animated version of selected pages. If the number and variety of pages seems too intimidating, consider taking one of our Guided Tours through the web site.

* Animation files can be large (average 350K bytes)
** Java Applet

SPEED REGIMES

Hypersonic Missions
Spacecraft Re-Entry
Hypersonic Cruise
High Supersonic
Supersonic

PROPULSION SYSTEMS

Interactive TBCC Simulator
Ramjet Propulsion
Scramjet Propulsion
Ramjet / Scramjet Thrust
Rocket Propulsion
Rocket Thrust
Gas Turbine Propulsion
Turbojet Thrust

RAMJETS & SCRAMJETS

Ramjet Parts
Brayton Cycle Analysis
Inlets
Inlet Performance
Combustors - Burners
Burner Thermodynamics
Nozzles
Nozzle Performance

ROCKET ENGINES

Liquid Rockets
Solid Rockets
Specific Impulse
Nozzle Design
Thrust Equations ... Interactive

AERODYNAMICS

Navier-Stokes Equations
Euler Equations
Speed of Sound ..Interactive**
Dynamic Pressure
Similarity Parameters ..Interactive**
Similarity Parameter Calculator

VISCOUS FLOW

Boundary Layer
Viscosity ..Interactive**
Reynolds Number ..Interactive**

COMPRESSIBLE FLOW

Mach Number ..Interactive**
Role of the Mach Number
Mach Calculator
Isentropic Flow...Interactive**
Isentropic Flow Calculator
Prandtl-Meyer Angle...Interactive**
Mach Angle...Interactive** ..
Compressible Mass Flow Rate...Interactive** ..
Compressible Area Ratio..Interactive**
Corrected Airflow per Unit Area..Interactive**
Centered Expansion Fan..Interactive**...

SHOCK WAVES

ShockModeler Interactive Simulator
Oblique Shocks..Interactive**
Oblique Shock Wave Simulator
Normal Shocks..Interactive**
Normal Shock Wave Calculator
Crossed Shocks..Interactive**...
Reflected Shock..Interactive**...

SCIENCE FUNDAMENTALS

Phases of Matter
Newton's Laws of Motion
Newton's First Law - Inertia
Newton's Second Law (F=ma)
Newton's Third Law (Action - Re-action)
Conservation of Mass
Conservation of Momentum
Conservation of Energy
Mass Flow Rate

MATH FUNDAMENTALS

Functions
Area
Volume
Scalars and Vectors
Comparing Two Scalars - Ratio
Comparing Two Vectors
Vector Components
Vector Addition
Trigonometry
Sine-Cosine-Tangent
Ratios in Triangles
Pythagorean Theorem..Interactive**

STATIC GASES

Animated Gas Lab...Animated
Gas Properties Definitions
Equation of State (Ideal Gas)
Specific Heats - cp and cv
Boyle's Law...Animated
Charles and Gay-Lussac's Law...Animated
Specific Volume
Kinetic Theory of Gases

THERMODYNAMICS

What is Thermodynamics?
Zeroth Law - Thermal Equilibrium
First Law - Internal Energy
Second Law - Entropy
Heat Transfer
Work
Work Done by a Gas
Enthalpy
Entropy of a Gas
Isentropic Compression

REAL GAS EFFECTS

Real Gas Effects..Interactive**
Specific Heats..Interactive**
Stagnation Temperature..Interactive**

THE ATMOSPHERE

AtmosModeler Interactive Simulator
Air Properties Definitions
Air Pressure
Air Temperature
Air Density
Earth Model - Imperial Units
Earth Model - Metric Units

Navigation..

NASA's Guide to Hypersonics Aerodynamics
Beginner's Guide Home Page

Table 1 List of webpages from NASA Hypersonic index used to evaluate ANSWER system's performance over text sources

Webpage Name	**Webpage Name**
Air Density	Navier-Stokes Equations
Air Temperature	Newton's Laws of Motion
Area	Newton's Second Law (F=ma)
Boundary Layer	Normal Shocks Interactive
Boyle's Law Animated	Nozzle Design
Burner Thermodynamics	Nozzle Performance
Centered Expansion Fan Interactive	Oblique Shocks Interactive
Charles and Gay-Lussac's Law Animated	Prandtl-Meyer Angle Interactive
Comparing Two Scalars - Ratio	Pythagorean Theorem Interactive
Compressible Area Ratio Interactive	Ramjet Scramjet Thrust
Compressible Mass Flow Rate Interactive	Ratios in Triangles
Conservation of Energy	Reynolds Number Interactive
Conservation of Mass	Rocket Thrust
Conservation of Momentum	Role of the Mach Number
Corrected Airflow per Unit Area Interactive	Second Law - Entropy
Dynamic Pressure	Similarity Parameters Interactive
Earth Model - Imperial Units	Sine-Cosine-Tangent
Earth Model - Metric Units	Solid Rockets
Enthalpy	Specific Heats Interactive
Entropy of a Gas	Specific Impulse
Equation of State (Ideal Gas)	Specific Volume
Euler Equations	Stagnation Temperature Interactive
First Law - Internal Energy	Thrust Equations Interactive
Functions	Trigonometry
Heat Transfer	Turbojet Thrust
Inlet Performance	Vector Addition
Isentropic Compression	Vector Components
Isentropic Flow Interactive	Viscosity Interactive
Liquid Rockets	Volume
Mach Angle Interactive	Work Done by a Gas
Mach Number Interactive	Work
Mass Flow Rate	

GE ASKE System Performance on Text Sources

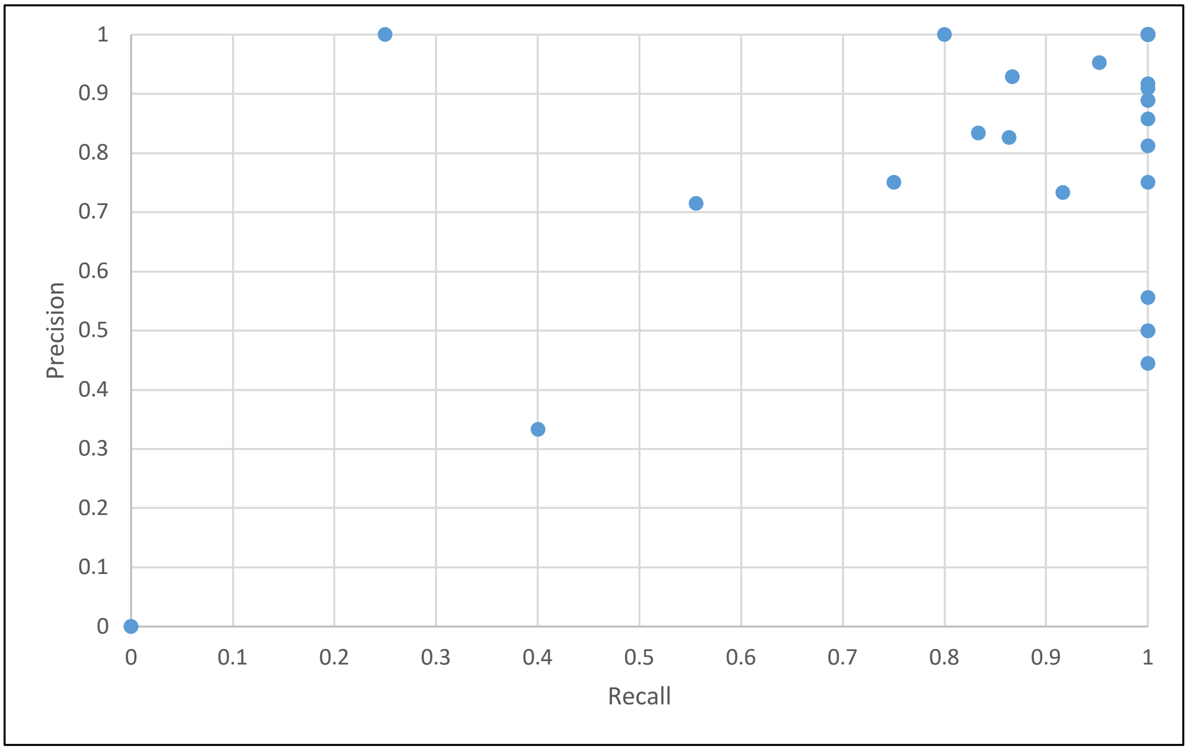


Figure 2: Precision Recall values for each of the 63 pages. Each dot represent a (recall value, precision value) pair for a given page

The GE ASKE System's performance over text sources was evaluated by measuring its ability to accurately extract equations from **63 pages** under the NASA Hypersonic Aerodynamics Index. We manually created a ground truth dataset of **358 equations** present in these 63 pages and computed two metrics: precision – which measures what fraction of the extracted equations are relevant/correct and recall – which measures the fraction of expected equations that were retrieved/extracted by the system. The **average precision** over the 63 pages is **0.89** and the **average recall** is **0.92** (*numbers closer to 1 indicate better performance*). Figure 2 gives a breakdown of the precision, recall values for the individual pages. As it can be seen from the graph, most pages have both a high precision and recall value simultaneously, indicating that the system not only extracted all expected equations but also most of the extracted equations were correct. In certain cases, the system also extracted additional incorrect/irrelevant equations which accounts for cases that have high recall values with lower precision values. Examples of incorrect, irrelevant and incomplete equations include "*v = .814 cubic meters/kilogram*", "*5^2 = 3^2 + 4^2*", and, "*x) = r * u * (del u / del x)*[1]".

[1] The complete accurate equation is - (del p / del x) = r * u * (del u / del x)

We also evaluated the system’s ability to convert extracted textual equations into executable Python methods. While there are 358 equations in our ground truth dataset, as mentioned above, the system extracts certain incorrect/irrelevant equations leading to 374 equations that the system attempted to translate into executable Python methods. Out of 374 equations, our text to python module failed to generate any python code for 98 of those. Some of those failed conversions can be attributed to the partial/incorrect extraction of equations from text. Given the nature of the equations that appear in textual format, the system often converts an equation into multiple python methods. For example, for the equation “*E2 - E1 = Q - W*”, text to python will generate two python methods – one with E2 as the return variable (*equation interpreted as “E2 = Q – W + E1”*) and one with E1 as the return variable (*equation interpreted as “E1 = -Q + W + E2”*). Overall, **from** the remaining **276 text equation strings**, **text to python generated 374 executable python methods** across 60 pages, of which **256 were deemed correct** (**77.7%**) by our subject-matter-expert.

Figure 3 Histogram showing the distribution of accuracy of generated python methods across different pages

Figure 3 gives a distribution of the accuracy of generated python methods across different pages. As we can see, accuracy for 39 out of 60 pages is 70% or greater with almost 50% of the pages having an accuracy of 90% or greater. Simple equations are the easiest to convert into functional Python code. Examples of such equations would be where there is a single variable on the left hand side and simple operations such as +, - , *, / on the right hand side. Slightly more complex equations, such as those having an operator on the left-hand side, were also handled without too many issues. The complicated ones were usually the ones that involved the exponent operations, e.g., ^ or e^. The tricky part was to backtrack from the ^ to figure out to what text segment the ^ operator was being applied. Especially complicated cases included having to deal with the unary negation operator in addition, e.g., e^-(1 + gamma) or e^-T. Currently, the translation from text to Python assumes everything is literal text. Therefore, expressions such as “exp” or “degrees” occurring in the text string will be translated as is, i.e., “exp” and “degrees”.

The code generator is essentially a parser that works with whatever it knows about. Therefore, whenever it encounters a new condition that it has not seen before, it will fail.

GE ASKE System Performance on Code Sources

Table 2 identifies the Java applets included on the Web site. Over the period of performance of this project, the number of broken links on the NASA Web site has increased. At the time we prepared this report, the Java applets listed in Table 2 were available for download. Note that to eliminate user-interface classes, the CodeExtractionModel (see Appendix B) specified ignoring the following classes: `Canvas`, `CardLayout`, `Graphics`, `Insets`, `Panel`, `Image`, `cem:Event`, `Choice`, `Button`, `Viewer`, `GridLayout`. Methods dealing with any of these classes or subclasses thereof were ignored during the extraction process. These extractions were done with a basic pre-existing domain ontology called SpeedOfSound. This ontology, and the ontologies that it imports, are listed in Appendix A.

Table 2: Java Applets Found on NASA Hypersonics Web Site and Extraction Results

Applet Class	Description	Methods in Class	Methods Extracted
Atmos.java	"Program to solve Standard Atmosphere Equations and using input velocity, compute the Mach number and dynamic pressure effects"	computeAtm os filter0 filter5 fiter3 init insets loadAlt loadGage loadOut setDefaults	computeAtm os filter0 filter3 filter5 Init loadAlt loadGage loadout setDefaults
CMFlow.java	"Program to calculate the mass flow rate given the Mach, area pressure and temperature"	comPute filter3 filter5 init insets loadOut setDefaults	comPute filter3 filter5 init loadOut setDefaults

Applet Class	Description	Methods in Class	Methods Extracted
Isentrop.java	"Program to perform one dimensional isentropic flow analysis from NACA 1135 Including Thermally perfect gas effects"	CAL_AQAS_LOOP CAL_ASQA CAL_DQDT CAL_DQDT_LOOP CAL_GAM CAL_GNU CAL_MFGNU CAL_PQPT CAL_PQPT_LOOP CAL_QQP CAL_QQP_LOOP CAL_SOS CAL_T CAL_TQTT_LOOP CAL_TT CAL_WCOR CAL_WCOR_LOOP comPute filter3 filter7 getAir getIsen getMach getMachArat getMachpm init insets loadOut setDefaults TQTT	CAL_AQAS_LOOP CAL_ASQA CAL_DQDT CAL_DQDT_LOOP CAL_GAM CAL_GNU CAL_MFGNU CAL_PQPT CAL_PQPT_LOOP CAL_QQP CAL_QQP_LOOP CAL_SOS CAL_T CAL_TQTT_LOOP CAL_TT CAL_WCOR CAL_WCOR_LOOP comPute filter3 filter7 getAir getIsen getMach getMachArat getMachpm init loadOut setDefaults TQTT

Applet Class	Description	Methods in Class	Methods Extracted
Mach.java	“Interactive Program to solve Standard Atmosphere Equations for mach and speed of sound Calorically imperfect gas modeled for hypersonics”	CAL_GAM CAL_SOS CAL_T CAL_TT CAL_TV computeMach filter0 filter5 fliter3 init insets loadInpt setDefaults TQTT	CAL_GAM CAL_SOS CAL_T CAL_TT CAL_TV computeMach filter0 filter3 filter5 init loadInpt setDefaults TQTT
Normal.java	“Interactive Program to solve Standard Atmosphere Equations and Normal shock relations Including Thermally Perfect Gas”	CAL_DNS CAL_GAM CAL_MNS CAL_PNS CAL_PQPT CAL_PTNS CAL_T CAL_TNS CAL_TT computeNormal filter0 filter3 filter5 filter9 init insets loadInpt setDefaults TQTT	CAL_DNS CAL_GAM CAL_MNS CAL_PNS CAL_PQPT CAL_PTNS CAL_T CAL_TNS CAL_TT computeNormal filter0 filter3 filter5 filter9 init loadInpt setDefaults TQTT

Applet Class	Description	Methods in Class	Methods Extracted
Shock.java	“Program to perform two dimensional analysis of supersonic flow past a wedge including oblique and normal shock conditions”	anlSing CAL_DNS CAL_GAM CAL_MNS CAL_PNS CAL_PQPT CAL_PTNS CAL_T CAL_TNS CAL_TT CAL_WAVE CAL_WAVE_TE ST comPute filter3 getAnglim getGeom getGeomNorm getGeomOblq getNorm getOblq getShkang init insets loadOut loadZero setDefaults TQTT	anlSling CAL_DNS CAL_GAM CAL_MNS CAL_PNS CAL_PQPT CAL_PTNS CAL_T CAL_TNS CAL_TT CAL_WAVE CAL_WAVE_TE ST comPute filter3 getAnglim getGeom getGeomNorm getGeomOblq getNorm getOblq getShkang init loadOut loadZero setDefaults TQTT
SPCal.java	“Interactive Program to solve for flow variables”	computeFlow filter0 filter3 filter5 filter9 init insets loadInpt loadOut setDefaults	computeFlow filter0 filter3 filter5 filter9 init loadInpt loadOut setDefaults

Applet Class	Description	Methods in Class	Methods Extracted
Turbo.java	“Program to perform turbomachinery design and analysis a) dry turbojet b) afterburning turbojet c) turbofan with separate nozzle d) ramjet”	(Solver)comPute (Solver)getFreeStream (Solver)getGeo (Solver)getPerform (Solver)getThermo (Solver)loadCF6 (Solver)loadF100 (Solver)loadJ85 (Solver)loadMine (Solver)loadRamj (Solver)myDesign (Solver)setDefaults (Solver)Solver filter0 filter1 filter3 getAir getCp getGama getMach getRayleighLoss init insets	(Solver)comPute (Solver)getFreeStream (Solver)getGeo (Solver)getPerform (Solver)getThermo (Solver)loadCF6 (Solver)loadF100 (Solver)loadJ85 (Solver)loadMine (Solver)loadRamj (Solver)myDesign (Solver)setDefaults (Solver)Solver filter0 filter1 filter3 getAir getCp getGama getMach getRayleightLoss init

Discussion

For each of these eight Java applets, the expected methods were extracted. In each case, an insets method was included in the applet but not extracted. This method has to do entirely with the user interface of the applet and was excluded from extraction by including it in the subclasses of ClassesToIgnore in the code extraction model defined in CodeExtractionModel.sadl (see Appendix B).

Many of the methods in these eight Java applets have implicit inputs and/or outputs. Usually these are in the form of class variables. If a class variable is used in a method as an input to a computation (right-hand side of an assignment operation of in a conditional), before it is computed (left-hand side of an operation), then it is an implicit input to the method. If it is computed before it is used, then it may be an implicit output. Rules in CodeExtractionModel.sadl (see Appendix B) are used to infer implicit inputs and outputs from the model extracted from the code’s parser-generated abstract syntax tree.

Because the Java code on the NASA Hypersonics Web Site has almost no comments, the extraction reported in Table 1 did not identify any semantic types for explicit method inputs, explicit method return values, or implicit inputs or outputs. However, in some cases it is possible to identify an applet with a particular HTML page and use that text to assist in the extraction from code. Two examples of this are given below.

Extraction from Mach.java with Locality Sound.txt

When extraction from Mach.java is done in conjunction with extraction from Sound.txt, the following additional information is extracted.

1. Implicit computeMach output “G” is given semantic type “gasConstant”
2. Implicit computeMach output “gama” is given semantic type “gasConstant”

Both of these are incorrect. “G” and “gama” both refer to the specific heat ratios of the gas.

Extraction from Turbo.java with Locality Isentrop.txt

When extraction from Turbo.java is done in conjunction with extraction from Isentrop.txt, the following additional information is extracted.

1. Implicit getFreeStream output “alt” is given semantic type “altitude”
2. Implicit getFreeStream output “u0” is given semantic type “velocity”
3. Implicit getFreeStream output “tsout” is given semantic type “gasConstant”
4. Implici setDefaults output “g0” is given semantic type “gasConstant”
5. Implicit getPerform output “g0” is given semantic type “gasConstant”
6. Implicit getPerform output “mfr” is given semantic type “Mass”

Of these, the first and second are correct, the third through fifth are incorrect, and the sixth is partially correct in that “mfr” is the mass flow ratio, so it is actually a dimensionless mass ratio. Note of these extractions are not specific in the most useful sense, e.g., altitude of what? Velocity of what?

Extraction of semantic meaning of variables in the absence of code comments is very difficult and our results do not show otherwise.

GE ASKE System Performance on Other Sources

To demonstrate generalization, we applied the extraction from code capability to a different domain, that of wind turbine design. We did extraction both without a pre-existing domain ontology and with a basic pre-existing domain ontology. This work was not completed in the timeframe covered by the M13 report. Therefore, the results of this effort will be included in the Final Report.

Computational Modeling in GE ASKE System: Comparison and Characteristics

We evaluated the performance of our model execution framework, KCHAIN, over models extracted from the NASA engine simulator code. The evaluation includes comparison of a TensorFlow and a NumPy-python versions of KCHAIN. At the time of this report, we have phased out the DBN execution framework and so we have not included it in the evaluation other than mentioning whether different characteristics or functionality is supported or not.

Characteristic	KHAIN with TensorFlow	KCHAIN with NumPy-Python	Other solutions	Comments
Just-in-time modeling (model is built and evaluated to obtain response to a certain query just-in-time)	Supported	Supported	Supported by GE's Dynamic Bayesian Network (DBN)	
Model build time (including REST API calls) - Average and std over 40 runs	7.9838 (±2.0225) seconds	0.0569 (±0.1360) seconds	N/A	Code generation of Python file is common to both approaches. TensorFlow (TF) approach additionally uses Autograph and AST transformation to create computational graph and saves it locally. This upfront investment is not suitable for just-in-time model creation and leads to 100x slow down.
Model evaluation time (including REST API calls) - Average and std over 40 runs	0.1366 (±0.0329) seconds	0.0179 (±0.0038) seconds	N/A	File I/O to load TensorFlow graph for each query evaluation is replaced with dynamic import of Python method into workspace leading to 10x speed up.
Gradient computation for local sensitivity	We use tf.gradient to compute gradients over nodes of computational graphs	We use AutoGrad [1](now part of JAX [2]) to compute gradients over arbitrary NumPy code at runtime.	Autodiff package for C++ [3]	
Branching statements	TF Autograph [4] is used to rewrite Python to create computational graph. This doesn't work if only one branch initializes a variable that was not defined prior to the branch statement.	Supported directly		

Characteristic	KHAIN with TensorFlow	KCHAIN with NumPy-Python	Other solutions	Comments
Loop blocks	Autograph doesn't work if loop variable or conditional variable is changed in certain ways within body of loop. This was addressed by AST rewriting to unroll the loop for a few iterations.	Supported directly		
Evaluating multiple input scenarios	Even though original model code doesn't support vectorized inputs, converting code to TF graph allows tensor computations to evaluate multiple inputs scenarios simultaneously. Although promising, this doesn't work if any input variable shows up as loop variable or in conditional expression in branching. For branching, this can be resolved with AST rewriting using vector conditionals (tf.where). No known direct solution for loop variable.	We use NumPy broadcasting to obtain tuples for each input scenario and evaluate each scenario independently in sequence or in parallel.		
Uncertainty Quantification	Intended to use TensorFlow probability, but documentation is poor and requires TF 2.x. Monte Carlo sampling of inputs can be conducted using capability of "Evaluating multiple input scenarios".	Monte Carlo sampling of inputs can be conducted using capability of "Evaluating multiple input scenarios", if input priors are available.	GE's DBN uses MCMC, HMC for UQ	This is a low risk extension. One challenge is knowing suitable parameter or input distributions.

Characteristic	KHAIN with TensorFlow	KCHAIN with NumPy-Python	Other solutions	Comments
Support for hybrid data-driven and physics-equation based modeling	Supported. TF supports data-driven models and physics equations. Hybrid models can be created with subgraphs as neural networks.	Supported. JAX is being developed by Google for deep learning with Python-NumPy coding. JAX is based on AutoGrad for gradient-based learning, which we are using for our models. Thus, hybrid models can be created with subgraphs as neural networks.		Challenge 1: to choose suitable architecture for the subgraph data-driven model. We can explore AutoML methods for that. Challenge 2a: to extract relations from text of "depends on" or "affects" that provide input and output variables for data-driven models as well as obtaining suitable datasets with similar inputs and outputs. Challenge 2b: to semantically align variables in dataframe with variables in text where relationship was identified.
Calibration or inverse queries	Gradient-based optimization can be used for model calibration. Bayesian optimization has been implemented to support inverse queries with intelligent sampling.	Gradient-based optimization can be used for model calibration. Bayesian optimization has been implemented to support inverse queries with intelligent sampling.		
Anytime modeling (computational model that evolves as knowledge graph evolves via curation)	Supported.	Not supported yet.		
Appending computational models	Supported. The append functionality connects existing computational graphs with new subgraphs based on I/O variable names.	Not supported yet. However, just-in-time composed models from knowledge base can be built and evaluated.		

Characteristic	KHAIN with TensorFlow	KCHAIN with NumPy-Python	Other solutions	Comments
Refinement or coarsening	Supported. If a method which uses existing name is added, we allow to overwrite that subgraph. The new subgraph can be a refinement or coarser alternative.	Not supported yet. However, just-in-time composed models from knowledge base with alternative methods can be built and evaluated.		
Handling default values for variables or constants	Default values to model I/O can be assigned during build steps and these values are stored in a dictionary to be used in evaluation if value is not assigned in the call.	Not supported yet as composed model from knowledge graph has the information of default values.		
Inform user that key variable for inference is missing	Supported. If value for a key variable is not provided during call for evaluation and is not available in default values, then we use knowledge of that missing variable to inform the user.	Supported. If value for some key variables is not provided during call for evaluation, then we inform the user of names of missing variables.		

Conclusions

The NASA Hypersonics Website provides a rich set of textual equations and associated text as well as a reasonable set of Java Applets for extraction from code. Unfortunately, the Java code has very few comments, making extraction of any semantic interpretation of method inputs and outputs, explicit or implicit, very difficult. Hybrid extraction from text and code can help but is still not adequate for this particular set of inputs. We had thought that alignment of equations in text and equations in code would greatly improve our capability and modified the Data Rights associated with the contract to enable this to be explored. Unfortunately, the envisioned approach for alignment did not work because the methods in the code, the granularity at which we were doing code analysis, did not align with the text equations. The general approach of the GE ASKE System would support extraction from code at a finer granularity than methods, making alignment with equations in text more feasible, but we did not have the time and resources to pursue this approach to extracting essential semantic information within the limits of the ASKE research contracts.

References

[1] https://github.com/HIPS/autograd

[2] https://github.com/google/jax

[3] https://github.com/autodiff/autodiff

[4] https://blog.tensorflow.org/2018/07/autograph-converts-python-into-tensorflow-graphs.html

Appendix A: Semantic Models Used for Extraction from NASA Java Applets and from Text

SpeedOfSound.sadl

```
/***********************************************************************
 * Note: This license has also been called the "New BSD License" or
 * "Modified BSD License". See also the 2-clause BSD License.
 *
 * Copyright © 2018-2019 - General Electric Company, All Rights Reserved
 *
 * Project: ANSWER, developed with the support of the Defense Advanced
 * Research Projects Agency (DARPA) under Agreement  No.  HR00111990006.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 *
 ***********************************************************************/
 uri "http://sadl.org/SpeedOfSound.sadl" alias sos.

 import "http://sadl.org/ScientificConcepts2.sadl".

 speedOfSound describes Gas with values of type Velocity.
 altitude describes Air with values of type Length.
 movesIn describes PhysicalObject with values of type Gas.
 mach describes PhysicalObject with a single value of type decimal.

 Gamma is a type of UnittedQuantity.
 gamma (note "treated as a calorically perfect gas") describes Gas with values of
type Gamma.
 gammaCal (note "treated as a calorically imperfect gas") describes Gas with values
of type Gamma.
```

```
 Theta (alias "theta") is a type of Constant.
 ThetaISU is a Theta with ^value 3056, with unit "Kelvin".
 ThetaImperial is a Theta with ^value 5500, with unit "Rankine".

 GasConstant is a type of UnittedQuantity.
 gasConstant (alias "R") describes Gas with values of type GasConstant.
 UniversalGasConstant is a type of Constant.
 UniversalGasConstantISU is a UniversalGasConstant with ^value 8.314, with unit
"J/mole-Kelvin".
 GasConstantAirISU is a UniversalGasConstant with ^value 0.286, with unit
"kJ/kg/degK".
 SpecificGasConstant is a type of GasConstant.
 MolarMass is a type of UnittedQuantity.
 molarMass describes Substance with values of type MolarMass.
 molarMass of Air always has value (a MolarMass MolarMassAirISU with ^value 28.96,
with unit "g/mole").
 molarMass of Air always has value (a MolarMass MolarMassAirImpoerial with ^value
28.96, with unit "lbm/lbmole").

  External machCalc(decimal sos (speedOfSound of some Air), // and sos < 10),
                             decimal sov (velocity of (a PhysicalObject movesIn the
Air)))
                             returns decimal (mach of the PhysicalObject):
"http://com.ge.research.darpa.aske.kchain.machcalc".

 External specificGasConstantEq(decimal Ru (UniversalGasConstant {"J/mole-Kelvin"}),
                                                    decimal M (molarMass of a Gas
{"g/mole"}))
                                                    returns decimal (gasConstant of
the Gas {"J/g-Kelvin"}):

       "http://com.ge.research.darpa.aske.kchain.specificGasConstantEq".

 External CAL_SOS(double T (temperature of a Gas {Kelvin, Rankine}),
                            double G (gamma of the Gas),
                            double R (gasConstant of the Gas {"g/mole",
"lbm/lbmole"}),
                            double Q (Theta {Kelvin, Rankine}),
                            string us (UnitSystem {Metric, Imperial}))
                            returns double (speedOfSound of the Gas {"m/sec",
"ft/sec"}): "http://com.ge.research.darpa.aske.kchain.CAL_SOS".

 External CAL_SOS_Imperial(double T (temperature of a Gas {Rankine}),
                            double G (gamma of the Gas),
                            double R (gasConstant of the Gas {"lbm/lbmole"}),
                            double Q (Theta {Rankine}))
                            returns double (speedOfSound of the Gas {"ft/sec"}):
"http://com.ge.research.darpa.aske.kchain.CAL_SOS".

 External CAL_GAM(double T (temperature of a Gas {Kelvin, Rankine}),
                            double G (gamma of the Gas),
                            double Q (Theta {Kelvin, Rankine}),
                            string us (UnitSystem {Metric, Imperial}))
                            returns double (gammaCal of the Gas):
                            "http://com.ge.research.darpa.aske.kchain.CAL_GAM".
 External tempFromAltitude (double alt (altitude of some Air {ft}))
                                                   returns double (temperature of the Air
{Rankine}):

       "http://com.ge.research.darpa.aske.kchain.tempFromAltitude".

// External computeMach(double alt (altitude of a PhysicalObject and the
PhysicalObject movesIn some Air and alt < 35000 {ft, m}),
//External computeMach(double alt (altitude of a PhysicalObject and the
PhysicalObject movesIn some Air and lessThan(alt, 35000) {ft, m}),
 External computeMach(double alt (altitude of a PhysicalObject and the PhysicalObject
movesIn some Air {ft, m}),
                                   double R (gasConstant of the Air {"lbm/lbmole",
"g/mole"}),
                                   double G (gamma of the Air),
                                   double Q (Theta {Kelvin, Rankine}),
                                   double vel (velocity of the PhysicalObject {"mph",
"km/hr"}),
                                   string us (UnitSystem {Metric, Imperial}))
                                   returns double (mach of the PhysicalObject):

"http://com.ge.research.darpa.aske.kchain.computeMach".

 Equation compMach(double sv (velocity of a PhysicalObject and the PhysicalObject
movesIn some Air {"ft/sec", mph, "m/sec", "km/hr"}),
                               double sos (speedOfSound of the Air {"ft/sec", mph,
"m/sec", "km/hr"})
                              ) returns double (mach of the PhysicalObject): return
sv/sos.

 compMach has expression (a Script with script "a+b", with language Java).

 AirGammaDefault is a Gamma.
 AirGammaDefault has ^value 1.4.
 gamma of Air has default AirGammaDefault.

// gamma of Air has default 1.4.
/*
      <j.0:Argument>
        <j.0:argType>http://www.w3.org/2001/XMLSchema#double</j.0:argType>
        <j.0:argName>alt</j.0:argName>
        <j.0:augmentedType rdf:parseType="Colleciton">
          <j.0:typeURI>http://wikidata/Q1234</j.0:typeURI>
               <j.0:triple>
                 <j.0:subject>X</j.0:subject>
                 <j.0:predicate>rdf:type</j.0:predicate>
                 <j.0:object>PhysicalObject</j.0:object>
               </j.0:triple>
               <j.0:triple>
                 <j.0:subject>X</j.0:subject>
                 <j.0:predicate>altitude</j.0:predicate>
                 <j.0:object>alt</j.0:object>
               </j.0:triple>
               <j.0:triple>
```

```
                  <j.0:subject>X</j.0:subject>
                  <j.0:predicate>movesIn</j.0:predicate>
                  <j.0:object>Y</j.0:object>
                </j.0:triple>
                <j.0:triple>
                  <j.0:subject>Y</j.0:subject>
                  <j.0:predicate>rdf:type</j.0:predicate>
                  <j.0:object>Air</j.0:object>
                </j.0:triple>
                <j.0:builtin
ref:"com.ge.research.sadl.jena.reasoner.builtin#lessThan">
                     <arg> alt</arg>
                     <arg 35000/>
                </j.0:builtin>
         </j.0:augmentedType>
       </j.0:Argument>

 */
// <X, rdf:type, PhysicalObject>, <X, altitude, alt>, <X, movesIn, Y>, <Y, rdf:type,
Air>

//Doc3 is a table [double alt (altitude of a PhysicalObject and the PhysicalObject
movesIn some Air {ft}),
//                             double vel (velocity of the PhysicalObject {mph}),
//                             double tt (temperature of the Air {R})]
//                             with data {[1,2,3],[4,5,6]}.
//Doc43 is a table [double alt (altitude of a PhysicalObject and the PhysicalObject
movesIn some Air {ft}),
//                             double vel (velocity of the PhysicalObject {mph}),
//                             double tt (temperature of the Air {R})]
//                             with data located at "http://someurl".
//MachTable is a type of table [double alt (altitude of a PhysicalObject and the
PhysicalObject movesIn some Air {ft}),
//                             double vel (velocity of a PhysicalObject {mph}),
//                             double tt (temperature of the Air {R})]
//                              with data {[1,2,3],[4,5,6]}. // this should be an
validation error
//MachTable2 is a class.
////Doc4 is a MachTable2 with data {[1,2,3],[4,5,6]}.

 External sosfromtemp(decimal t) returns decimal:
"http://com.ge.research.darpa.aske.kchain.sosairfromtemp".
 External sosfromtemp2(decimal t (temperature of some Air {"Kelvin", Rankine}))
      returns decimal (speedOfSound of the Air {"m/sec","ft/sec"}):
      "http://com.ge.research.darpa.aske.kchain.sosairfromtemp".

 External sosfromalt(decimal alt) returns decimal:
"http://com.ge.research.darpa.aske.kchain.sosairfromalt".
 // with implied properties made explicit
 External sosfromalt2(decimal alt (^value of altitude of some Air))
      returns decimal (^value of speedOfSound of the Air):
"http://com.ge.research.darpa.aske.kchain.sosairfromalt".

// External machCalc(decimal sos (speedOfSound of some Air and sos < 10 and
lessThan(sos, gasConstant of the Air)),
//                           decimal sov (velocity of (a PhysicalObject movesIn the
Air)))
//                           returns decimal (mach of the PhysicalObject):
"http://com.ge.research.darpa.aske.kchain.machcalc".

 Rule SpeedOfSoundInAirGivenTemperature:
   if air is some Air and t is temperature of air and
   sosair is  cgSC(sosfromtemp, t)
   then speedOfSound of air is sosair.

 Rule SpeedOfSoundInAirGivenTemperatureK:
   if air is some Air and t is temperature of air and
   unit of t is "Kelvin" and
   sosair is  cg(sosfromtemp, t)
   then there exists (a Velocity with ^value sosair, with unit "m/sec") and
   speedOfSound of air is the Velocity.

 Rule SpeedOfSoundInAirGivenAltitude:
   if air is some Air and alt is altitude of air and
   lunit is unit of alt and
   vunit is strConcat(lunit, "/sec") and
   sosair is  cg(sosfromalt, alt)
   then there exists (a Velocity with ^value sosair, with unit vunit) and
   speedOfSound of air is the Velocity.

 Equation specificGasConstantEq2(decimal Ru (UniversalGasConstant {"J/mole-Kelvin"}),
                                                    decimal M (molarMass of a Gas
{"g/mole"}))

                                                    returns decimal (gasConstant of
the Gas {"J/g-Kelvin"}):

                                                    ^value of Ru/M.
```

ScientificConcepts2.sadl

```
/**********************************************************************
 * Note: This license has also been called the "New BSD License" or
 * "Modified BSD License". See also the 2-clause BSD License.
 *
 * Copyright © 2018-2019 - General Electric Company, All Rights Reserved
 *
 * Project: ANSWER, developed with the support of the Defense Advanced
 * Research Projects Agency (DARPA) under Agreement  No.  HR00111990006.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 *
 **********************************************************************/
uri "http://sadl.org/ScientificConcepts2.sadl" alias scicncpts2.

primitiveData is a type of {decimal or boolean or string}.

Constant is a type of UnittedQuantity.
UnitSystem is a class, must be one of {Metric, Imperial}.

Derivative is a type of ScientificConcept,
      described by derivativeOf with a single value of type ScientificConcept,
      described by withRespectTo with a single value of type class.

Time is a type of UnittedQuantity.
Temperature /* (alias "temperature")*/ is a type of UnittedQuantity.
Length is a type of UnittedQuantity.
Position is a type of UnittedQuantity,
      described by x-coordinate with values of type Length,
      described by y-coordinate with values of type Length,
       described by z-coordinate with values of type Length,
       described by ^time with values of type Time.
Mass is a type of UnittedQuantity.

Volume is a type of UnittedQuantity.
Density is a type of UnittedQuantity.

PhysicalThing is a class,
       described by mass with values of type Mass,
       described by volume with values of type Volume,
       described by density with values of type Density,
       described by temperature with values of type Temperature,
       described by staticTemperature with values of type Temperature.

External densityEq(decimal m (mass of a PhysicalThing),
                      decimal v (volume of the PhysicalThing))
                      returns decimal (density of the PhysicalThing):
                      "http://com.ge.research.darpa.aske.kchain.densityEq".

PhysicalObject is a type of PhysicalThing,
       described by position with values of type Position.

Substance is a type of PhysicalThing.

Gas is a type of Substance.
Air is a type of Gas.

Velocity is a type of {UnittedQuantity and Derivative}.
derivativeOf of Velocity only has values of type Position.
withRespectTo of Velocity only has values of type Time.
velocity (alias "speed") describes PhysicalObject with values of type Velocity.

Acceleration is a type of {UnittedQuantity and Derivative}.
derivativeOf of Acceleration only has values of type Velocity.
withRespectTo of Acceleration  only has values of type Time.
acceleration describes PhysicalObject with values of type Acceleration.

Momentum is a type of {UnittedQuantity and Derivative}.
momentum describes PhysicalThing with values of type Momentum.
Rule momentumOfPhysicalThing:
       if o is a PhysicalObject with velocity v and
             p is a Momentum with ^value (^value of mass of o * ^value of velocity of
o),
                    with unit unitResolver("*", unit of mass of o, unit of velocity
of o)
       then momentum of o is p.

Force is a type of {UnittedQuantity and Derivative}.
derivativeOf of Force only has values of type Momentum.
withRespectTo of Force only has values of type Time.
force describes PhysicalObject with values of type Force.

External unitResolver(string operation, string u, ...)
       returns string: "http://sadl.org/unitSelector".
```

```
External cg(ExternalEquation eq, primitiveData arg, ...) returns primitiveData:
"http://com.ge.research.darpa.aske.cg".
External cgSC(ExternalEquation eq, primitiveData arg, ...) returns ScientificConcept:
"http://com.ge.research.darpa.aske.cgsc".
```

SadlImplicitModel.sadl (automatically imported by every SADL model)

```
/*********************************************************************
 * Note: This license has also been called the "New BSD License" or
 * "Modified BSD License". See also the 2-clause BSD License.
 *
 * Copyright © 2018-2019 - General Electric Company, All Rights Reserved
 *
 * Project: ANSWER, developed with the support of the Defense Advanced
 * Research Projects Agency (DARPA) under Agreement  No.  HR00111990006.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 *
 *********************************************************************/
uri "http://sadl.org/sadlimplicitmodel" alias sadlimplicitmodel.
/****** The content of this model is automatically included in every SADL model
******/

impliedProperty is a type of annotation.
expandedProperty is a type of annotation.
ScientificConcept is a class.
UnittedQuantity is a type of ScientificConcept,
	described by ^value with values of type decimal,
	described by stddev with values of type decimal,
	described by unit with values of type string.
```

```
DataDescriptor is a class, described by localDescriptorName (note "If this
DataDescriptor is associated with a named parameter, this is the name") with a single
value of type string,
       described by dataType (note "the simple data type, e.g., float") with a single
value of type anyURI,
       described by specifiedUnits (note "the array of possible units") with a single
value of type string List,
       described by augmentedType (note "ties the DataDescriptor to the semantic
domain model") with values of type AugmentedType,
       described by descriptorVariable (note "This identifies the GPVariable, if any,
in the AugmentedType which is associated with this DataDescriptor").
dataType of DataDescriptor has at most 1 value.
descriptorVariable of DataDescriptor has at most 1 value.

Language is a class.
{Java, Python, Text, OtherLanguage} are instances of Language.
Script is a class, described by language with a single value of type Language,
       described by script with a single value of type string.
^Equation is a class,
       described by expression with values of type Script.
arguments describes ^Equation with a single value of type DataDescriptor List.
returnTypes describes ^Equation with a single value of type DataDescriptor List.

ExternalEquation is a type of ^Equation,
       described by externalURI with a single value of type anyURI,
       described by externalURL with values of type anyURI.

AugmentedType is a class.
augTypeUnits describes AugmentedType with a single value of type string List.
SemanticType (note "allows direct specification of the semantic type of an argument")
is a type of AugmentedType,
       described by semType with a single value of type class.
GraphPattern is a class.
{TriplePattern, FunctionPattern} are types of GraphPattern.
gpSubject describes TriplePattern.
gpPredicate describes TriplePattern.
gpObject describes TriplePattern.
builtin describes FunctionPattern with a single value of type ^Equation.
GPAtom is a class.
{GPVariable, GPLiteralValue, GPResource} are types of GPAtom.
gpVariableName describes GPVariable with a single value of type string.
gpLiteralValue describes GPLiteralValue with values of type data.
argValues (note "values of arguments to the built-in") describes FunctionPattern with
a single value of type GPAtom List.

SemanticConstraint (note "used to identify necessary patterns in semantic domain
terms") is a type of AugmentedType,
       described by constraints with a single value of type GraphPattern List.
ThisArgument (note "allows reference to self within an Argument's constraints") is a
DataDescriptor.

anyDataType (note "union of all relevant data types") is a type of {decimal or
boolean or string or date or dateTime or anyURI}.
DataTableRow is a class,
       described by rowValues with a single value of type anyDataType List.
```

```
DataTable is a class,
       described by columnDescriptors with a single value of type DataDescriptor
List,
       described by dataContent with a single value of type DataTableRow List,
       described by dataLocation with a single value of type anyURI.

^Rule is a class.
NamedQuery is a class.
derivedFrom (note "for use with named structures (Equations, NamedQueries, ...),
where it has special SADL syntax")
       is a type of annotation.


/****** The following content comes from the Dialog implicit model fragment provider
******/
dependsOn is a property with values of type class.
UnittedQuantity has impliedProperty ^value.

IntializerMethod is a type of ExternalEquation,
       described by initializes with values of type class.
initializerKeyword is a type of annotation.
IntializerMethod has initializerKeyword "load", has initializerKeyword "initialize".

minValue describes DataDescriptor.
maxValue describes DataDescriptor.

ImplicitDataDescriptor is a type of DataDescriptor,
       described by declaration with values of type Script.
implicitInput describes ^Equation with values of type ImplicitDataDescriptor.
implicitOutput describes ^Equation with values of type ImplicitDataDescriptor.

{Python-TF, Python-NumPy} are instances of Language.
matchingClass is a type of annotation.
matchingProperty is a type of annotation.
```

Appendix B: The Code Extraction Model

```
/***********************************************************************
 * Note: This license has also been called the "New BSD License" or
 * "Modified BSD License". See also the 2-clause BSD License.
 *
 * Copyright © 2018-2019 - General Electric Company, All Rights Reserved
 *
 * Project: ANSWER, developed with the support of the Defense Advanced
 * Research Projects Agency (DARPA) under Agreement  No.  HR00111990006.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the copyright holder nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 *
 ***********************************************************************/
uri "http://sadl.org/CodeExtractionModel.sadl" alias cem.

// This is the code extraction meta-model
CodeElement is a class described by beginsAt with a single value of type int,
       described by endsAt with a single value of type int.

CodeBlock is a type of CodeElement,
       described by serialization with a single value of type string,
       described by comment with values of type Comment,
       described by containedIn with values of type CodeBlock.

{Class, Method, ConditionalBlock, LoopBlock} are types of CodeBlock.

cmArguments describes Method with a single value of type CodeVariable List.
cmReturnTypes describes Method with a single value of type string List.
cmSemanticReturnTypes describes Method with a single value of type string List.
doesComputation describes Method with a single value of type boolean.
incompleteInformation describes Method with a single value of type boolean.
```

```
calls describes Method with values of type MethodCall.
deadCode describes Method with values of type boolean.
isCalled describes Method with values of type boolean.
ExternalMethod is a type of Method.
Constructor is a type of Method.
Rule CisM: if x is a Constructor then x is a Method.

// The reference to a CodeVariable can be its definition (Defined),
//     an assignment or reassignment (Reassigned), or just a reference
//     in the right-hand side of an assignment or a conditional (Used)
Usage is a class, must be one of {Defined, Used, Reassigned}.

Reference  is a type of CodeElement
       described by firstRef (note "first reference in this CodeBlock")
              with a single value of type boolean
       described by codeBlock with a single value of type CodeBlock
       described by usage with values of type Usage
       described by cem:input (note "CodeVariable is an input to codeBlock
CodeBlock")
              with a single value of type boolean
       described by output (note "CodeVariable is an output of codeBlock CodeBlock")
              with a single value of type boolean
       described by isImplicit (note "the input or output of this reference is
implicit (inferred), not explicit")
              with a single value of type boolean
       described by setterArgument (note "is this variable input to a setter?") with
a single value of type boolean
       described by comment with values of type Comment.

MethodCall is a type of CodeElement
       described by codeBlock with a single value of type CodeBlock
       described by inputMapping with values of type InputMapping,
       described by returnedMapping with values of type OutputMapping.
MethodCallMapping is a class,
       described by callingVariable with a single value of type CodeVariable,
       described by calledVariable with a single value of type CodeVariable.
{InputMapping, OutputMapping} are types of MethodCallMapping.

Comment (note "CodeBlock and Reference can have a Comment") is a type of CodeElement
       described by commentContent with a single value of type string.

// what about Constant also? Note something maybe an input and then gets reassigned
// Constant could be defined in terms of being set by equations that only involve
Constants
// Constants could also relate variables used in different equations as being same
CodeVariable  is a type of CodeElement,
       described by varName with a single value of type string,
       described by varType with a single value of type string,
       described by semanticVarType with a single value of type string,
       described by quantityKind (note "this should be qudt:QuantityKind") with a
single value of type ScientificConcept,
       described by reference with values of type Reference.

{ClassField, MethodArgument, MethodVariable, ConstantVariable} are types of
CodeVariable.
```

```
constantValue describes ConstantVariable with values of type UnittedQuantity.
//External findFirstLocation (CodeVariable cv) returns int: "http://ToBeImplemented".

Rule Transitive
if inst is a cls and
   cls is a type of CodeVariable
then inst is a CodeVariable.

Rule Transitive2
if inst is a cls and
   cls is a type of CodeBlock
then inst is a CodeBlock.

Rule FindFirstRef
if c is a CodeVariable and
   ref is reference of c and
   ref has codeBlock cb and
   l is beginsAt of ref and
   minLoc = min(c, reference, r, r, codeBlock, cb, r, beginsAt) and
   l = minLoc
then firstRef of ref is true
//    and print(c, " at ", minLoc, " is first reference.")
.

Rule ImplicitInput
if cb is a CodeBlock and
   ref has codeBlock cb and
   ref has firstRef true and
   ref has usage Used
   and cv has reference ref
//   and ref has beginsAt loc
then input of ref is true and isImplicit of ref is true
//    and print(cb, cv, loc, " implicit input")
.

Rule ImplicitOutput
if cb is a CodeBlock and
   ref has codeBlock cb and
   ref has firstRef true and
   ref has usage Reassigned
   and cv has reference ref
   and noValue(cv, reference, ref2, ref2, codeBlock, cb, ref2, usage, Defined)
//   and ref has beginsAt loc
then output of ref is true and isImplicit of ref is true
//    and print(cb, cv, loc, " implicit output")
.
Rule IsCalled
if    m is a Method and
      mc is a MethodCall and
      mc codeBlock m
then m isCalled true.

Rule DeadCode
if  m1 is a Method and
```

```
        m2 is a Method and
        m1 != m2 and
        m1 calls mc and
        mc codeBlock m2 and
        mc returnedMapping rm and
        rm callingVariable cv and
        noValue(cv, reference, ref, ref, usage, Used)
then deadCode of m2 is true.

ClassesToIgnore is a type of Class.
{Canvas, CardLayout, Graphics, Insets, Panel, Image, cem:Event, Choice, Button,
        Viewer, GridLayout, Math, Double, Float, String
} are types of ClassesToIgnore.

Ask ImplicitMethodInputs: "select distinct ?m ?cv ?vt ?vn where {?r <isImplicit> true
. ?r <http://sadl.org/CodeExtractionModel.sadl#input> true .
        ?r <codeBlock> ?m . ?cv <reference> ?r . ?cv <varType> ?vt . ?cv <varName>
?vn} order by ?m ?vn".
Ask ImplicitMethodOutputs: "select distinct ?m ?cv ?vt ?vn where {?r <isImplicit>
true . ?r <http://sadl.org/CodeExtractionModel.sadl#output> true .
        ?r <codeBlock> ?m . ?cv <reference> ?r . ?cv <varType> ?vt. ?cv <varName> ?vn}
order by ?m ?vn".
Ask MethodsDoingComputation: "select ?m where {?m <doesComputation> true}".
Ask MethodCalls: "select ?m ?mcalled where {?m <calls> ?mc . ?mc <codeBlock>
?mcalled} order by ?m ?mcalled".Ask VarComment: "select ?cmntcontent ?eln ?usage
where { ? <http://sadl.org/CodeExtractionModel.sadl#reference> ?ref .
        ?ref <http://sadl.org/CodeExtractionModel.sadl#endsAt> ?eln .
        ?ref <http://sadl.org/CodeExtractionModel.sadl#usage> ?usage .
        ?cmnt <rdf:type> <http://sadl.org/CodeExtractionModel.sadl#Comment> .
        ?cmnt <http://sadl.org/CodeExtractionModel.sadl#endsAt> ?eln .
        ?cmnt <http://sadl.org/CodeExtractionModel.sadl#commentContent> ?cmntcontent}
order by ?eln".
```