Unified Decomposition-Aggregation (UDA) Rules: Dynamic, Schematic, Novel Axioms

Unified Decomposition-Aggregation (UDA) Rules: Dynamic, Schematic, Novel Axioms

Alan Bundy and Kwabena Nuamah

School of Informatics, University of Edinburgh, UK
{A.Bundy,K.Nuamah}@ed.ac.uk

Abstract. We introduce *Unified Decomposition-Aggregation (UDA) rules.* They are a family of axiom schemata that are instantiated at run-time to add new axioms to a logical theory. These new axioms are implications, whose preconditions will be constructed from an analysis of the goal to be proved and the theory in which it is to be proved. We illustrate their application to query answering using the FRANK system.

Keywords: UDA rules, query answering, hybrid reasoning

1 Introduction

Unified Decomposition-Aggregation (UDA) rules[1] were invented as part of the FRANK (Functional Reasoner for Acquiring New Knowledge) query answering system [10, 5]. They have the potential for wider application. We define UDA rules in Definition 3.https://www.overleaf.com/project/61f904969bf6e6fd899f9441

UDA rules decompose a goal into a set of subgoals. The answers returned by the subgoals are then aggregated into an answer to the original goal using an *aggregation function.* This decomposition process can recur. What distinguishes a UDA rule from other implications is that it dynamically computes the number and form of the subgoals. For instance, to answer the query "*What will be the population of the UK in 2030?*" regression over past census data is used via a *temporal UDA rule.* The subgoals might represent the sub-queries "*What will be the population of the UK in* t_i*?*", where each t_i is a year for which the UK population can be retrieved from prior census data sources. Regression is then applied to these previous population numbers and the resulting function is extrapolated to the year 2030. Another distinction is that the answers to queries are incrementally both constructed and evaluated by aggregation functions during their propagation from leaves to the root of the inference graph. A comparison between FRANK's inference process and that of traditional automated reasoning systems is given in [5].

> *The main novel contribution of this paper is to demonstrate that axiomatic theories can be dynamically constructed by adding correct new axioms to a theory on an as-needed basis.*

[1] Previously called decomposition rules.

There are many kinds of UDA rules: parent goals can be decomposed into child sub-goals in many different ways and the corresponding child answers can be aggregated in many different ways to return a value to their parent goal. There is not a 1-1 correspondence between decompositions and aggregations, there are many legitimate combinations depending on the original query, retrieved data and user preferences, e.g., speed *vs* accuracy.

Instantiation of a schematic UDA rule may be incremental. For instance, the choice of aggregation function may be delayed, e.g., the kind of regression to be used may depend on the available data. This means that the precise form of a UDA rule may not be known until inference is complete.

To illustrate the use of UDA rules, suppose the query is:

> *What will be the population in 2030 of the African country that will have the largest GDP in 2025?.*

First, a *partition UDA rule* can be used to formulate a query to predict the 2025 GDP of each African country and return the country with the largest one. Temporal decomposition can be used to make this prediction for each country. Then, temporal decomposition will be used again to predict the 2030 population of the selected country.

Partition rules can also be used to estimate the total population of a continent by summing the populations of its individual countries. Or, the cost of a compound object might be estimated by summing the costs of each of its components.

A precursor to UDA rules can be found in the first author's Mecho project [3], which solved mechanics problems stated in English. Part of the solution process consisted of instantiating the laws of Physics, such as $F = m.a$, where m is the mass of an object, a its acceleration and F the sum of the forces acting on it. Consider the task of instantiating F. This required Mecho to partition this sum of forces into each of the individual forces, e.g., gravitational attraction, tension in a string, friction from an inclined plane, etc. The partition UDA rule would have been ideal for this task, but since we had not then invented such rules, Mecho dealt with it in a more ad hoc way. The potential for additional applications of UDA rules can be found in §7.

These examples illustrate the potentially wide applicability of UDA rules, especially as automated reasoning broadens its remit beyond the traditional axiomatic theories of pure mathematics and formal methods, to reason about the wider environment.

2 Association Lists

FRANK uses UDA rules to construct an inference graph in which each decomposition of a goal into subgoals represents an AND branch in the graph. The leaves of the graph are subgoals that can be instantiated by matching facts from web-based knowledge sources. It uses a diverse and dynamically chosen set of knowledge sources whose facts are represented in a diverse number of formalisms.

In order to combine these facts, they are translated into a common formalism: association lists, abbreviated as *alists*[2] [10]. Alists are defined in Definition 1.

Definition 1 (Alist) *An* alist *is a set of pairs* $\{\langle A_i, a_i\rangle | 1 \leq i \leq n\}$*, where each* A_i *is an attribute and* a_i *is its value. This will sometimes be written as* $\{\langle A_1, a_1\rangle, \ldots, \langle A_n, a_n\rangle\}$ *or abbreviated as* $\mathcal{A}$*.*

- *One attribute must be* $Predicate$*. This allows an alist:*

$$\{\langle Predicate, P\rangle, \langle A_1, a_1\rangle, \ldots, \langle A_n, a_n\rangle\}$$

  to be represented as the typed relation $P(a_1, \ldots, a_n)$ *where its type is* $P : A_1 \times \ldots \times A_n \mapsto Bool$*, i.e., the* A_i *attributes are interpreted as the types of their values.*
- *Typical attributes are* $Subject$*,* $Object$*,* $Predicate$*,* $Time$*, etc, abbreviated as* s*,* o*,* p *and* t*. Values can be names of entities, numbers, functions, etc.*
- *We will use the notation* $\mathcal{A}(t)$ *to indicate that* $\mathcal{A}$ *contains a distinguished term* t *at some unspecified argument position, e.g., an attribute or its value.*
- *We also need to interpret Alists as functions from some of its values to others. For this purpose we use Hilbert's Epsilon Operator*[3]*, written:* $\epsilon x.\ \mathcal{A}(x)$ *where the alist* $\mathcal{A}$ *contains the variable* x*. This is read as the value of* x *such that* $\mathcal{A}(x)$ *is true. If there is more than one such value, an arbitrary choice is made. If there is no such value then* $\mathcal{A}(x)$ *is false, so it cannot be the child alist of a successful decomposition, and can, therefore, be ignored.*

Further details about the semantics of alists are discussed in the second author's thesis [10].

3 Variables in Alists

Alists can represent goals by specifying a variable as the value for at least one of its attributes. A goal will be satisfied during inference by instantiating its variable to a concrete value. Instantiation occurs when a leaf alist is matched to a fact in a knowledge base. The values of variable are then propagated through the inference graph from the leaves to the root alist. At each stage the values of the child alists are aggregated to give a value for the parent alist. Aggregation is performed by the *aggregation functions* of the UDA rules used. Aggregation functions range from the identity function, through various arithmetic functions (e.g., Σ, max, min) to statistical operations, such as regression. FRANK's range of statistical methods has recently been significantly extended [8].

Definition 2 describes the different kinds of variables used in an alist as defined in [10].

[2] See https://en.wikipedia.org/wiki/Association_list (last accessed: 02-02-2022). Alists are not lists but sets, but the 'alist' terminology has, unfortunately, become standard.

[3] https://en.wikipedia.org/wiki/Epsilon_calculus accessed on 02.02.2022

Definition 2 (Kinds of Variables)

Projection Variables: Those variables in a child alist whose values are to be projected to its parent. *They become operation arguments of the aggregation function in the parent alist. They are prefixed with a ?, e.g., ?x denotes a projection variable. In general, a child alist may have several projection variables, so we use vector notation to denote them all, e.g.,* $?\vec{x}$.

Auxiliary Variables: Those variables in a child alist whose values are used locally, but which are not projected to its parents. *They are prefixed with a* \$*, e.g.,* \$$x$ *denotes an auxiliary variable. In general, an alist may have several auxiliary variables, so we use vector notation to denote them all, e.g.,* $\$\vec{x}$.

Operation Variables: Those variables that are used as arguments for an $\mathcal{A}$'s *aggregation function* h. *An operation variable must be either a projection or an auxiliary variable, so we omit any prefix. They must exist as an attribute value in* $\mathcal{A}$.

Projection variables are instantiated to values that are aggregated from child alists and projected to their parents or by matching against facts in a knowledge base. The value projected to the root of the inference graph becomes the answer to the original query.

For instance, consider the question from §1:

> *What will be the population in 2030 of the African country that will have the largest GDP in 2025?.*

The inference graph that FRANK generates to answer this query will consist of several nodes labelled by alists. Two such alists are (1) and (2). Alist (1) represents the sub-query where FRANK predicts the GDP $?g$ of Ghana in 2025. This alist is one of many generated for the different countries in Africa in order to find the one with the largest predicted GDP in 2025. Alist (2) represents the sub-query where FRANK predicts the population $?p$ of country \$$c$ in 2030.

$$\{\langle p, gdp\rangle, \langle s, Ghana\rangle, \langle o, ?g\rangle, \langle t, 2025\rangle\} \quad (1)$$

$$\{\langle p, population\rangle, \langle s, \$c\rangle, \langle o, ?p\rangle, \langle t, 2030\rangle\} \quad (2)$$

where p is the predicate, s the subject, o the object and t the time. \$$c$ is an auxiliary variable that will be instantiated to the country with the largest GDP. $?g$ and $?p$ are projection variables whose values will be propagated to their parents. In the case of $?g$ it will then become an auxiliary variable to the *max* function, to determine the country with the largest GDP. In the case of $?p$ it will be the population of country \$$c$. Note that, in one of the children of (2), \$$c$ will have been a projection variable $?c$, to be propagated to alist (2).

Each alist has an aggregation function attribute h with a value h_τ, say. This function h_τ is applied to the projection variables of the child alists to instantiate the projection variable of the parent. This aggregation function is associated with the UDA rule on the AND branch connecting the parent alist to its children. The

aggregation operation requires each alist to be regarded as a function from the projection variables of the children to the projection variable of the parent. As described in §2, we use the Hilbert Epsilon operation ϵ to convert alists from relations to functions for this purpose.

4 UDA rules

UDA rules are implications applied by backwards reasoning to a parent alist to create a set of child alists, i.e., the logical implication is from children to parent. The size of this set and the anatomy of its members is described by: the type τ of the rule, the parent alist that it is applied to and the environment in which it is applied. τ is incrementally defined during the inference process. For instance, when a choice is made to use a partition rule, τ will be instantiated to $part(\upsilon)$, where *part* refers to the partition rule and υ is a variable that will eventually be instantiated to specify the type of aggregation. When this function is chosen to be the summation of the values of the child values, the rule type, $part(\upsilon)$, will be further instantiated to $part(\Sigma)$. In general, τ will be different for each UDA rule application. For instance, the children of a partition UDA rule will depend on the available partitions of the parent, e.g., by *partOf* relations. The children of a temporal UDA rule will depend on the available values of the parent's predicate for earlier values of its time attribute, e.g., by $before$ relations. The general form of a UDA rule is given in Definition 3. Their correctness is defined in §4.3 and their application is illustrated in §5.

Definition 3 (UDA rule) *A UDA rule is an implication of the form:*

$$\begin{aligned} &Decompose(\mathcal{A}(\vec{x}), \tau) = [\mathcal{A}_j(?\vec{x}_j) | 1 \leq j \leq m] \\ &\quad \Longrightarrow \mathcal{A}(\vec{h_\tau}(\epsilon ?\vec{x}_1.\, \mathcal{A}_1(?\vec{x}_1), \ldots, \epsilon ?\vec{x}_m.\, \mathcal{A}_m(?\vec{x}_m))) \end{aligned}$$

where:

- *Decompose is a function that takes the parent alist $\mathcal{A}(\vec{x})$ and the type of decomposition τ and returns a list of m child alists $[\mathcal{A}_j(?\vec{x}_j) | 1 \leq j \leq m]$.*
- *The various alists differ only in the attribute values singled out as their arguments, e.g., $\vec{x}$ in $\mathcal{A}(\vec{x})$. To suppress clutter, their attributes and other values are not indicated, but are identical in each alist.*
- *$\vec{h_\tau}$ is the aggregation function that takes the values $\epsilon ?\vec{x}_j.\, \mathcal{A}_j(?\vec{x}_j)$ assigned to the projection variables $?\vec{x}_j$ of the child alists $\mathcal{A}_j(?\vec{x}_j)$ and calculates an aggregated value:*

  $$\vec{h_\tau}(\epsilon ?\vec{x}_1.\, \mathcal{A}_1(?\vec{x}_1), \ldots, \epsilon ?\vec{x}_m.\, \mathcal{A}_m(?\vec{x}_m))$$

  to be projected back to the parent alist $\mathcal{A}(\vec{x})$ as the value(s) of the variable(s) $\vec{x}$. Note that the $?\vec{x}_j$ are projection variables as their values will be projected back to their parent alist $\mathcal{A}(\vec{x})$. The $\vec{x}$ may or may not be projection variables.

– $\vec{h_\tau}$ *is selected during inference as an operation to aggregate the values returned by the new child alists. Note that the choice of aggregation function might depend on the values to be aggregated, the query intent, context and the decomposition. For instance, the choice of an aggregation function for prediction during temporal decomposition will depend on, say, the number of data points available and their underlying statistical distribution. So, the choice may be left as a variable to be instantiated only after the UDA rule has been applied.*
– $[\mathcal{A}_j | 1 \leq j \leq m]$ *is a form of list composition, that we have invented, which is analogous to set comprehension. Lists are used, rather than sets, because argument order may matter*[4] *for the aggregation function* $\vec{h_\tau}$.
– *Vector notation is used for the variables* $\vec{x_j}$ *and* $\vec{x}$*, and for the aggregation function* $\vec{h_\tau}$ *because more than one variable and/or function may be involved in a rule.*
– *Note that the logical implication of UDA rules is from right to left: the values of the projection variables of the child alists determine the values of the operands of the parent alist. But FRANK builds the inference graph by applying the UDA rules left to right, i.e., from the goal alist to the leaf node alists. The projection variables of the leaf alist are then instantiated by matching them against facts in the knowledge sources.*

The above UDA rule can be represented graphically as shown in Figure 1.

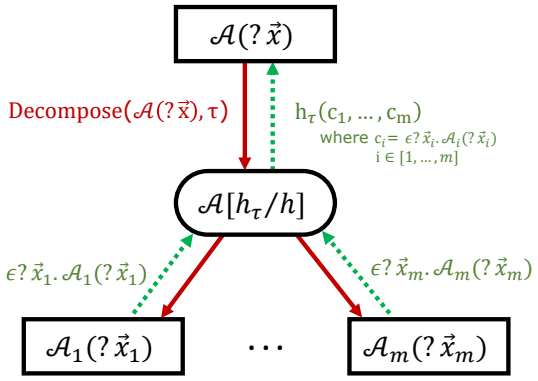

Fig. 1. A graphical representation of a UDA rule. It shows a single step of inference in FRANK and is applied recursively to construct inference graphs. Solid red arrows show the direction of decomposition while the dashed green ones show the direction of aggregation during upward propagation of projection variable instantiations from children to their parent.

[4] Although not for any of the examples in this paper.

Another novel feature of UDA rules is that they define two kinds of inference: firstly, the replacement of parent goals with child sub-goals via decomposition and secondly, the propagation of answers from leaves to root via aggregation. Unlike most automated reasoning, where variables are just instantiated to compound terms, aggregation evaluates these compound terms to return values, e.g., numbers, countries.

4.1 The New Axiom

We can describe the new axiom being added by a UDA rule using the alternative first-order logic (FOL) notation introduced in §2. Let P be the value of the compulsory *Predicate* attribute of $\mathcal{A}(\vec{x})$ and P_j its value in $\mathcal{A}_j(?\vec{x}_j)$ for $1 \leq j \leq m$. Then the new axiom is:

$$P_1(?\vec{x}_1) \wedge \ldots \wedge P_m(?\vec{x}_m) \implies P(\vec{h_\tau}(?\vec{x}_1, \ldots, ?\vec{x}_m)) \tag{3}$$

For the sake of readability, we have omitted any additional arguments of P and the P_j.

If the UDA rule was used to add this axiom directly into the logical theory, then any FOL theorem prover, e.g., resolution [1], could be used to derive the required answer to the query.

4.2 Representing Uncertainty

FRANK applies both arithmetic and statistical aggregation functions, e.g., summation and regression. Statistical aggregation introduces uncertainty into the reasoning, so we cannot, in general, demand logical correctness of the axioms that UDA rules introduce into a theory. There is also noise in the knowledge sources. So, the validity of FRANK's inference is statistical as well as logical. It, therefore, returns an error interval with its conclusions [12, 10].

Noise in the answer is assumed to be a Gaussian[5]. For quantitative queries, FRANK uses the mean as the answer and the coefficient of variation (*cov*), which is the standard deviation normalised by the mean, as the measure of uncertainty. The *cov* can be interpreted as an error interval around the answer. To calculate the answer's *cov*, each $\mathcal{A}$ and each $\mathcal{A}_j$ is given an additional *cov* attribute/value pair and an uncertainty propagation function, similar to $\vec{h_\tau}$, is used to propagate the uncertainties of the child alists $\mathcal{A}_j$s to the uncertainty of their parent $\mathcal{A}$.

4.3 The Correctness of the UDA rules

We can associate a *correctness property* with each UDA rule. In the case of rules that use a statistical aggregation function, the correctness property is also statistical. The proofs of these correctness properties would be conducted in a theory of sets, and the definition of the corresponding aggregation functions. Their automation is currently future work.

[5] Or similar, depending on the statistical methods used.

The Non-Statistical Case We consider, first, the non-statistical case. Consider the partition rule for decomposing a compound object into a partition of sub-objects and then summing the values that they return. We name this rule $part(\Sigma)$. The definition of *Decompose* for the $part(\Sigma)$ rule and its corresponding correctness property are given in Definition 4.

Definition 4 (Partition Correctness Property) *Consider the following definition of Decompose.*

$$Decompose(\mathcal{A}(O,x), part(\Sigma)) = [\mathcal{A}_j(O_j, x_j) \mid partOf(O_j, O)]$$

where $partOf(O_j, O)$ *means that* O_j *is an* immediate *part of compound object* O*, i.e., there is no* O_j *such that it is a part of a part. We assume that the call to* $partOf$ *returns all such parts, so the* O_j*s form a partition. We have distinguished two attribute values in* $\mathcal{A}(O,x)$ *and the* $\mathcal{A}_j(O_j, x_j)$*: the objects* O *and their properties* x*. Otherwise, the alists* $\mathcal{A}$ *and the* $\mathcal{A}_j$ *are identical.*

Note that we have dropped the vector notation as, in this rule, the x *and* x_j *are singleton variables.*

The $part(\Sigma)$ *rule correctness property is now:*

$$x = \Sigma([x_j \mid \mathcal{A}_j(O_j, x_j) \wedge partOf(O_j, O)]) \implies \mathcal{A}(O,x) \tag{4}$$

where the x_j *are formed into a list to which* Σ *is applied and the result* x *gives a value that makes* $\mathcal{A}(O,x)$ *true.*

Sometimes the partition rule is used not to sum the child scores, but to find a maximum, minimum or apply another arithmetic operation. Different correctness properties are needed for these cases, such as:

$$x = max([x_j \mid \mathcal{A}_j(O_j, x_j) \wedge partOf(O_j, O)]) \implies \mathcal{A}(O,x)$$
$$x = min([x_j \mid \mathcal{A}_j(O_j, x_j) \wedge partOf(O_j, O)]) \implies \mathcal{A}(O,x)$$

The Statistical Case We now consider the statistical case, illustrated by the regression aggregation function, *regress*, used by the temporal rule, which we name the $temp(reg)$ rule [6]. Definition 5 defines *Decompose* for the $temp(reg)$ rule and its corresponding correctness property.

Definition 5 (Temporal Correctness Property)

$$Decompose(\mathcal{A}(T,y), temp(reg)) = [\mathcal{A}_j(T_j, y_j) \mid before(T_j, T)]$$

where $before(T_j, T)$ *means that time* T_j *occurs before time* T*. We have distinguished two attribute values in the alists: the times* T *and the alists' property's values* y *at the corresponding times. We use* y *instead of* x*, as, traditionally,*

[6] Note that the temporal rule can also use non-statistical aggregation functions, e.g., $temp(max)$ could be used to find the maximum value of a property among a set of times

the x-axis would be time and the y-axis is the value of the property over time. Otherwise, the alists $\mathcal{A}$ and the $\mathcal{A}_j$ are identical.

$$\rho : \langle f, st \rangle = regress([y_j \mid \mathcal{A}_j(T_j, y_j) \wedge before(T_j, T)])$$
$$\implies \exists y.\ y - sd(T) \leq F(T) \leq y + sd(T) \wedge \mathcal{A}(T, y)$$

regress returns the pair of the function f that it creates together with the first standard deviation function sd. $f(T)$ and $sd(T)$ extrapolate the functions f and sd to the time T. y is the value that makes $\mathcal{A}(T, y)$ true. The correctness property asserts that $f(T)$ lies within the error interval, i.e., within one standard deviation of y. ρ is the probability that $f(T)$ lies within this interval, which can be calculated using the `68-95-99.7` *rule*[7]*, which gives the value* 0.6827*. $\langle T_j, y_j \rangle$ are the points used in the regression.*

FRANK's inference graph is constructed by applying correct UDA rules, but there are many branching points, so search is controlled by heuristics. For instance, FRANK prefers to consult knowledge sources containing facts with low uncertainty and to use decompositions that result in a representative list of child alists and that respect user preferences [13].

5 Worked Example

To illustrate the use of these UDA rules in FRANK, consider the following query.

What will be the population of Africa in 2030?

This can be represented as the following alist:

$$\{\langle p, population \rangle, \langle s, Africa \rangle, \langle o, ?x \rangle, \langle t, 2030 \rangle\} \tag{5}$$

where the 4 attributes p, s, o, t mean *predicate*, *subject*, *object* and *time*, respectively and $?x$ is a projection variable.

To answer this query FRANK can use two UDA rules: $part(\Sigma)$ and $temp(reg)$. $part(\Sigma)$ is used to partition $Africa$ into its constituent countries and $temp(reg)$ is used to estimate the 2030 populations for each country, which are then summed using the Σ aggregation function of $part(\Sigma)$.

FRANK's GUI interface for this query is given in Figure 2. FRANK's search strategy is to apply correct UDA rules respecting the heuristics outlined in §4.3. One possible inference path proceeds as follows.

1. FRANK tries to match alist (5) to various knowledge sources, but finds no match because Africa's population for 2030 is not yet known.
2. It then makes an unsuccessful attempt to apply $temp(reg)$ to alist (5), which fails due to the lack of census records for the whole of Africa.

[7] https://en.wikipedia.org/wiki/68-95-99.7_rule accessed on 12.5.22

3. Instead, it makes a successful attempt to apply $part(\Sigma)$ to alist (5) which creates the following 54 child alists:

$$\{\langle p, population\rangle, \langle s, Algeria\rangle, \langle o, ?x_1\rangle, \langle t, 2030\rangle\}$$
$$\ldots \{\langle p, population\rangle, \langle s, Zimbabwe\rangle, \langle o, ?x_{54}\rangle, \langle t, 2030\rangle\}$$

4. FRANK tries to match each of these alists to various knowledge sources, but finds no matches because population data for 2030 is not yet known.
5. It then makes successful attempts to apply $temp(reg)$ to each of these 54 child alists. The *regress* aggregation function returns a population growth function and a standard deviation. Both are extrapolated to 2030 and the resulting 2030 population estimates and their standard deviations are propagated back to their shared parent alist.
6. $part(\Sigma)$'s aggregation function, Σ, is used to sum the 54 population estimates. Their standard deviations are turned into *cov*s so that they can be combined both together and with a *cov* for the regression. The resulting *cov* is also propagated to the parent alist to provide the uncertainty estimate for the population estimate.
7. Both the population and uncertainty estimates are returned to the user.

6 Related Work

We have not found anything in the literature directly related to UDA rules. For instance, although axioms of the form (3) are commonly used in automated reasoning [1], they are usually provided manually by human experts, rather than automatically generated to meet the needs of the current (sub)goal.

There are also logics in which uncertainty measures are associated with axioms [7, 9], but these measures are usually probabilities or similar. We found probabilities to be unsuitable as an uncertainty measure for quantitative answers. The probability that a particular number is the true answer to a query is usually 0. For instance, the precise annual population of a country is inherently uncertain, given the long period and whether tourists, illegal immigrants, babies being born, people in a vegetative state, people in hiding, etc should be or can be counted. The range given by an error interval is a much better uncertainty measure. Probabilities are, however, an appropriate measure for *qualitative* answers, e.g., multi-hop information retrieval [6]. We are exploring their use in FRANK, including situations where a query requires mixing probabilities with error intervals, e.g., consider the example in §1, where the *cov*s on each African country's GDP must be propagated into the probability that a particular country has the largest GDP.

The way in which UDA rules dynamically construct the inference search space, e.g., by determining the branching rate of a rule, is reminiscent of *proof planning* [2], whereby meta-level rules are used to pre-plan the shape of a proof[8].

[8] We are grateful to an anonymous reviewer for pointing out this analogy and suggesting that we discuss it here.

The main difference is that proof planning uses a two-level inference process: meta-level and object-level rules, whereas UDA rules are dynamically constructed object-level rules.

An alternative to an alist representation might be one based on functions. For instance, the $part(\Sigma)$ rule might be represented as:

$$F(y) = \Sigma(\{F_j(y_j)|partOf(y, y_j) \wedge 1 \leq j \leq m\})$$

where, using the functional interpretation of alists, F is associated with $\mathcal{A}$, F_j with $\mathcal{A}_j$, y is the compound object and the y_j are its parts. The projection variables are now the outputs of the F and F_j functions. This functional representation might be used, for instance, in a functional programming implementation of FRANK.

7 Future Work

We plan to develop some theories in which the UDA rule correctness properties can be proved, initially manually, but ultimately automatically by FRANK. We also plan to develop our correctness properties for particular UDA rules into a general theory of UDA rule correctness.

We are considering the use of a choice-style UDA rule for meta-level inference, i.e., to automate FRANK's engineering decisions, such as which knowledge sources and reasoning methods to use to solve the current query. This will assist us in our vision of whole-system explanations of AI systems [4, 11]. By representing these choices in a declarative form, FRANK's explanation system will be able to describe not only the object-level inference it uses to answer a query, but also the meta-level inference it used to construct the object-level inference process from knowledge sources and reasoning methods.

We are experimenting with a probabilistic uncertainty measure for qualitative queries. These uncertainty measures need to be interleaved during inference as a quantitative value propagated from a child alist may be converted into a quantitative value propagated by its parent and vice versa, e.g., *cov*s on each African country's GDP may be propagated into the probability that a particular country has the largest GDP.

8 Conclusion

We have introduced the concept of UDA rules. They are implication schemata that are instantiated during the inference process to create new axioms to be added to a logical theory and then used in a proof. These new axioms depend on: the type τ of the rule, the parent alist that it is applied to, the original goal and the theory in which it is applied. They are useful whenever the number of preconditions of the new axiom depends on these factors, e.g., the number of components of a compound object, the number of available values of a time varying attribute, the forces acting on an object, the number of applicable reasoning methods and the number of relevant knowledge sources.

UDA rules have been implemented in the FRANK query answering system. We have proposed correctness properties for the different kinds of UDA rules. Some of FRANK's UDA rules have statistical aggregation functions, such as regression. The correctness properties for these rules give only a probability that FRANK's answers lie within the first standard deviation of the true answer. The knowledge bases from which FRANK draws its facts are also potentially noisy. Given these two sources of uncertainty, FRANK returns both an estimated answer and an error interval.

Acknowledgements

Thanks to Nicholas Ferguson, Thomas Fletcher, Xue Li, Ruqui Zhu and five anonymous reviewers for feedback on an earlier draft. This research has been supported by Huawei grant CIENG4721/LSC.

References

1. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, Volume 1, vol. I, chap. 2, pp. 19–199. Elsevier (2001)
2. Bundy, A.: A Science of Reasoning, pp. 178–198. MIT Press (1991)
3. Bundy, A., Byrd, L., Luger, G., Mellish, C., Milne, R., Palmer, M.: Solving mechanics problems using meta-level inference. In: Buchanan, B.G. (ed.) Proceedings of IJCAI-79. pp. 1017–1027. International Joint Conference on Artificial Intelligence (1979)
4. Bundy, A., Nuamah, K.: Combining deductive and statistical explanations in the FRANK query answering system. In: Gong, Z., Li, X., Oguducu, S.G. (eds.) Proceedings of the 12th IEEE International Conference on Big Knowledge (ICBK). IEEE, Auckland, New Zealand (December 2021)
5. Bundy, A., Nuamah, K., Lucas, C.: Automated reasoning in the age of the internet. In: 13th International Conference on Artificial Intelligence and Symbolic Computation. vol. LNAI 11110, pp. 3–18. Springer, Cham (8 2018). https://doi.org/10.1007/978-3-319-99957-9_1, invited Talk
6. Das, R., Godbole, A., Kavarthapu, D., Gong, Z., Singhal, A., Yu, M., Guo, X., Gao, T., Zamani, H., Zaheer, M., et al.: Multi-step entity-centric information retrieval for multi-hop question answering. In: Proceedings of the 2nd Workshop on Machine Reading for Question Answering. pp. 113–118 (2019)
7. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted boolean formulas. Theory and Practice of Logic Programming **15**(3), 358–401 (2015)
8. Fletcher, T., Bundy, A., Nuamah, K.: Statistics automation in a query-answering system, submitted
9. Nilsson, N.J.: Probabilistic logic. Artificial intelligence **28**(1), 71–87 (1986)
10. Nuamah, K.: Functional inferences over heterogeneous data (2018), unpublished Ph.D. Dissertation, University of Edinburgh
11. Nuamah, K.: Deep algorithmic question answering: Towards a compositionally hybrid AI for algorithmic reasoning. In: Workshop on Knowledge Representation for Hybrid and Compositional AI (2021)

12. Nuamah, K., Bundy, A.: Calculating error bars on inferences from web data. In: SAI Intelligent Systems Conference (IntelliSys). pp. 618–640. Springer, Cham (11 2018). https://doi.org/10.1007/978-3-030-01057-7_48
13. Nuamah, K., Bundy, A., Jia, Y.: A context mechanism for an inference-based question answering system. In: AAAI2021 Workshop on CSKGs, Feb. vol. 8 (2021)

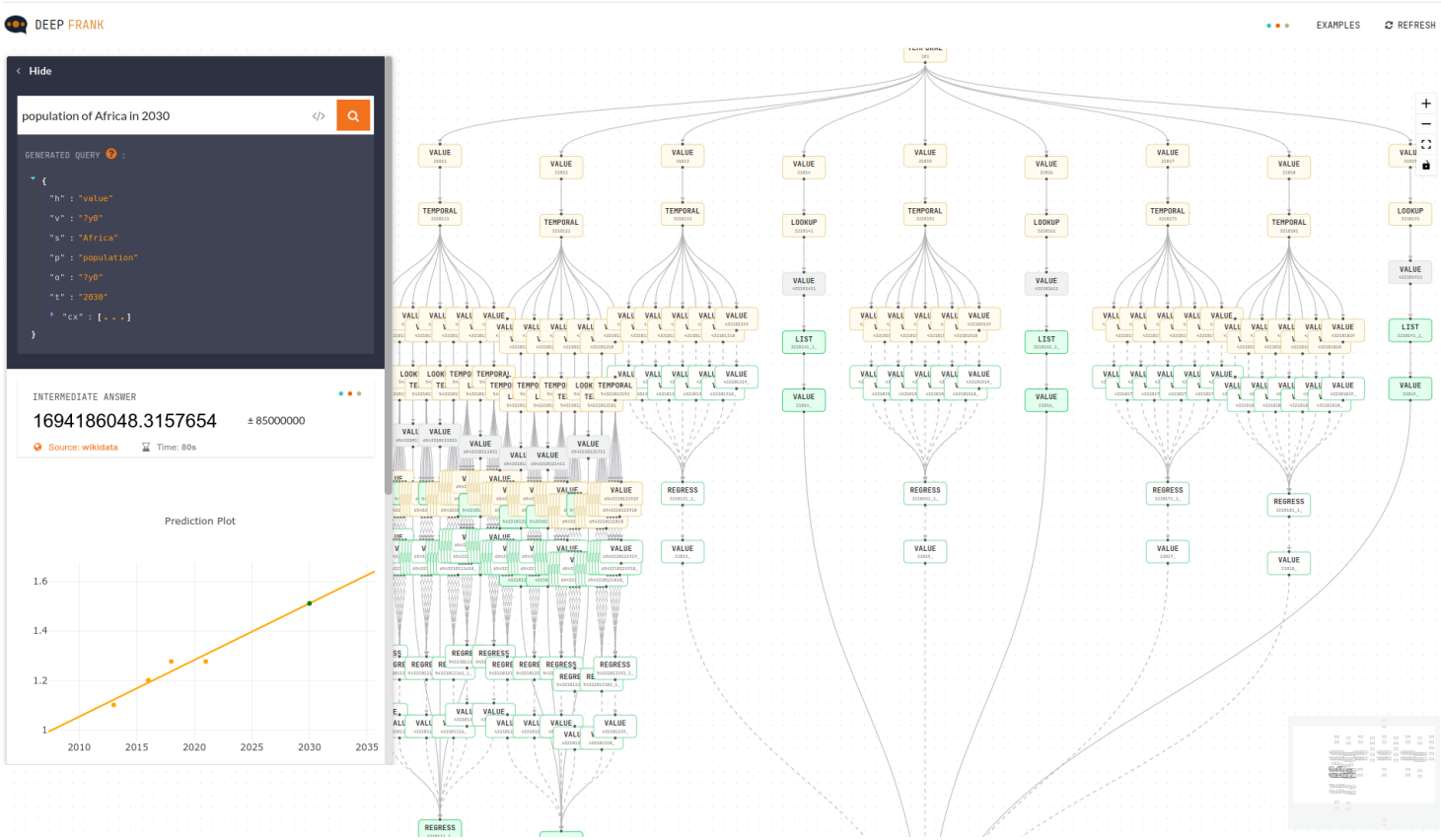

Fig. 2. The web browser-based user interface for FRANK. On the left side is shown the user's question, the generated alist query, the intermediate answer, error intervals on the answer and the knowledge sources used. On the right is shown the corresponding inference graph.