From QED to X-Logic

following the lead of Leibniz

Roger Bishop Jones
rbj@rbjones.com

2007/06/12

Outline

Leibniz (1646 -1716)

- .. an intellectual innovator of the first rank, his work included:
  - A philosophical system which was "unusually coherent and complete". (Russell)
  - The integral and differential calculus.
  - A calculating machine.
  - Many grand and unrealistic schemes, including one for the automation of reason.
- He was "an encyclopedic savant whose philosophy was nourished by the study of all the sciences and in turn inspired all of his scientific discoveries" (Couturat)
- "Mathematicians have as much need to be philosophers as philosophers have to be mathematicians." (Leibniz)
- Inspired:
  - Frege's *Begriffsschrift*,
  - Russell's work on *Principia* and other aspects of his philosophy
  - (indirectly) Carnap's philosophical programme, including the formalisation of physics.

Leibniz's Grand Projects

- Universal Mathematics
- The General Science
- Universal Encyclopedia
- Universal Language
- Academies and Journals
- Universal Characteristic and Calculus Ratiocinator
  - all knowledge formally expressible using the universal characteristic
  - all disputes resolvable by computation using the calculus ratiocinator
- Should we take this (characteristic and calculus) seriously?
  1. as an alternative to "the singularity"
  2. to establish the domain in which machines can be authoritative and make sure that they are reliable where they can be
  3. as promoting systematisation and simplification
  4. ...

"ARG philosophy"

Some pragmatic philosophical positions which appear to be endemic to the Cambridge Automated Reasoning Group.

1. Pragmatic Foundationalism
   - There are good practical reasons for doing formal deduction in the context of a "foundation system" (Gordon, Paulson ...)
     (i.e. a system in which one can work by *conservative extension*).
   - Getting deductive inference right is not enough, need to ensure that the context is coherent.
2. Deduction alone tells us nothing about anything physical (Fetzer!), ...
   but we can reason about physical reality using abstract models (Cohn).
3. To get your proof system sound its a good idea to have a semantics.
   A natural place to give the semantics of a foundation system is in set theory (e.g. Pitts and Arthan),
   Hence:
   - For establishing deductive soundness, abstract semantics suffices...
   - even if we want to reason about the real world.
4. A correct computation is as good as a proof (LCF paradigm?).
5. Interactive tools get you further (than plain automation), ... and we have ways of making them safe (LCF paradigm)

Why Should YOU Care (about Leibniz's project)?

Think:

- Deductive Design Automation (instead of EDA, etc.)
- Computer Aided Deductive Design (instead of CADE)

These are economic motivators.
Our contemporary architecture for the Leibniz Project (let's call it X-Logic) seeks to reduce to a bare minimum the cost of "Deductive Rigour".

- This is done by:
  1. Assimilating proof and computation.
  2. Separating rigour and assurance.
  3. Placing no lower bound on assurance.
- "Deductive Rigour" requires (of computation as proof):
  1. Every program has a formal specification (with which it complies!).
  2. Programs are only relied upon to meet their specification.
- and admits:
  - any correct program as an inference rule, with
  - assurance of correctness of conclusion no worse than assurance of correctness of program

Universal Formal Language

Is a Universal Formal Language possible or desirable?

- In Principle:
  - Concerned only with domains in which deductive reasoning is possible.
  - Any domain can be covered using abstract models
  - Hence, need language universal only for abstract modelling.
  - Set theory is therefore a contender.
    ... but "semantic regress" is a problem.
- In Practice:
  - Pluralism desirable
  - The development of new material (i.e. adding definitions) may constitute the definition of a new language.
  - Single extensible language for Mathematics.
  - "Semantic regress" isn't a problem.
- Coherent interworking in a pluralistic context is realisable through common semantics foundations.

Calculus Ratiocinator

- We have universal calculating machines.
- We know there is no decision procedure, but ...
  ... we do have near complete formal deductive systems.
- Computational feasibility limits the range of soluble problems,
  ... but formalisation is worthwhile and realisable,
  ... with automation sufficient to facilitate formal derivation of theory.

Feasible Goal

- An architecture supporting distributed co-operative large scale formalisation,
- Existing proof tools easily incorporated into the architecture.
- ... enabling continued development of automated deductive problem solving.

What is Architecture?

An architecture should supply:

- The most important high-level requirements
- A high level model of the system sufficient to expose...
- A rationale showing how or why the system will meet the requirements,

Requirements

- Soundness
  - An architecture for automated deductive reasoning, must of course be *deductive*.
  - So what does that mean?

    > *A demonstration is deductive if it is accomplished exclusively by the use of sound rules of inference.*

  - Therefore:
    - the system must be sound,
    - languages should have a well-defined *abstract* semantics.
- Functionality, Performance, Assurance
  - Flexible and known assurance.
  - Cost of rigour minimised.
  - Admit existing methods and tools.
  - Good for formal mathematics and formalisable science and engineering.
  - Economic driver / long term goal: The Automation of Design.

A Formal Architectural Model

This model addresses just some of the most important requirements.

These are key to the aim of "reducing the cost of rigour".

The model:

- is document oriented
- is multilingual
- admits inference by correct computation
- involves an assurance calculus

It is presented as the definition of an elementary top-level language in which proofs at lower levels can be sewn together while tracking the assurance level of the results.

A Model

$$
\begin{aligned}
&\textit{types } \textit{URI} = \textit{string} \\
&\quad \textit{document} = \textit{URI} \\
&\quad \textit{language} = \textit{URI} \\
&\quad \textit{program} = \textit{URI} \\
&\quad \textit{authority} = \textit{URI}
\end{aligned}
$$

Here:

- URI stands for Universal Resource Identifier but may also include a digest of the resource referred to.
- A document may be any static representation of information which has (or can be given) a *propositional* semantics, e.g. an XML document, a snapshot of a view of a database.
- Think of a language as encompassing a context, and a document as potentially creating a new context (like a theory).
- Any program for which a specification is “known” counts as a sound inference rule.
- We do not seek absolute assurance, sets of authorities serve as assurance levels.

Sentences and Judgements

The language contains sentences which express the claims that:

1. certain documents are true documents of particular languages
2. a certain program satisfies a specification formulated as soundness with respect to given lists of input and output languages
3. a specified list of output documents was computed by a program from a list of input documents

```
datatype sentence =
  TrueDocs "document list" "language list"
| ProgSpec program "language list" "language list"
| Compute  program "document list" "document list"
```

Judgements either assert sentences or endorse authorities:

```
types stamp = nat
```

```
datatype judgement =
    Assert stamp "authority set" sentence |
    Endorse authority "authority set"
```

Sentence Interpretations

To interpret a sentence we need to know:

1. the content of the documents
2. the semantics of the languages
3. the functions computed by the programs

types
 document_map = "*document* ⇒ *string*"
 language_map = "*language* ⇒ *string set*"
 program_map = "*program* ⇒ (*string list* ⇒ *string list*)"

record Sinterp =
  docmap :: *document_map*
  langmap:: *language_map*
  progmap:: *program_map*

True Sentences

```
consts
    truedoclist :: "[Sinterp, document list, language list] ⇒ bool"
primrec
   "truedoclist i (h_d#t_d) l_l = (case l_l of
      []          ⇒ False |
      (h_l#t_l) ⇒ (docmap i h_d) ∈ (langmap i h_l) & truedoclist i t_d t_l)"
   "truedoclist i [] l_l = (case l_l of [] ⇒ True | (h_l#t_l) ⇒ False)"

consts
   truesen :: "Sinterp ⇒ sentence ⇒ bool"

primrec
   "truesen i (TrueDocs dl ll)      = truedoclist i dl ll"
   "truesen i (ProgSpec p ill oll) = (∀ idl . truedoclist i idl ill
                                        --> truedoclist i (progmap i p idl) oll)"
   "truesen i (Compute p idl odl)  = (odl = progmap i p idl)"
```

Judgement Interpretations

- A judgement is true if infallibility of its authorities implies the truth of its sentence.
- To determine truth we therefore need to know what judgements have been asserted by each authority.

types
judgement_map = "*authority* ⇒ *judgement set*"

record Jinterp =
sinterp::*Sinterp*
judgemap::*judgement_map*

Infallibility and Truth - I

types
inftest = "*authority* ⇒ *Jinterp* ⇒ *bool*"
truthtest = "*judgement* ⇒ *Jinterp* ⇒ *bool*"

constdefs
authrel :: "*judgement_map* ⇒ (*authority* ∗ *authority*)*set*"
"*authrel jm* == *rtrancl* {*p*. ∃*as*. *snd p* ∈ *as*
& *Endorse* (*fst p*) *as* ∈ *jm* (*fst p*)}"

basett :: "*truthtest*"
"*basett j ji* == *case j of* (*Endorse a as*) ⇒ *True* |
(*Assert n as s*) ⇒ *truesen* (*sinterp ji*) *s*"

itrec :: "*nat* ⇒ (*inftest* ∗ *truthtest*) ⇒ *inftest*"
"*itrec n tsts auth ji* == ∀*a*. (*auth*, *a*) ∈ *authrel* (*judgemap ji*)
−−> (∀*j*. *j* ∈ *judgemap ji a* & *jstamp j* < *n* −−> *snd tsts j ji*)"

ttrec :: "(*inftest* ∗ *truthtest*) ⇒ *truthtest*"
"*ttrec tsts j ji* == (∀*auth*. *auth* ∈ (*jauths j*) −−> (*fst tsts*) *auth ji*)
−−> *basett j ji*"

Infallibility and Truth - II

consts
ittt :: "*nat* $\Rightarrow$ (*inftest* $*$ *truthtest*)"

primrec
"*ittt* *0* = (λ*x* *y* *z*. *True*), *basett*)"
"*ittt* (*Suc* *n*) = (*itrec* (*Suc* *n*) (*ittt* *n*), *ttrec* (*ittt* *n*))"

constdefs
hitherto_infallible :: "*nat* $\Rightarrow$ *authority* $\Rightarrow$ *Jinterp* $\Rightarrow$ *bool*"
"*hitherto_infallible* *n* == *fst* (*ittt* *n*)"

true_judgement :: "*judgement* $\Rightarrow$ *Jinterp* $\Rightarrow$ *bool*"
"*true_judgement* *j* == *snd* (*ittt* (*jstamp* *j*)) *j*"

Critical Requirements Formalised

Soundness

Inference preserves truth.

constdefs

sound ::"(*judgement set* ⇒ *judgement set*) ⇒ *bool*"
"*sound f* == ∀*ji js*. (∀*j*. *j*:*js* −−> *true_judgement j ji*)
−−> (∀*j*. *j*:(*f js*) −−> *true_judgement j ji*)"

Assurance

Lowly assured premises neither yield not influence highly assured conclusions.

constdefs

filter ::"*authority set* ⇒ *judgement set* ⇒ *judgement set*"
"*filter as js* == {*j* . *jauths j* ⊆ *Image* (*authr js*) *as*}"

assured ::"(*judgement set* ⇒ *judgement set*) ⇒ *bool*"
"*assured f* == ∀*js1 js2 as*. *filter as js1* = *filter as js2* −−>
filter as (*f js1*) = *filter as* (*f js2*)"

Outstanding Problems

This model addresses only a very few of the critical requirememts, and more modelling is needed in the following areas:

- The problem of consistency.
- Semantics and semantic definitions.
- Context
- Integrity

Foundations

- Semantics and consistency are crucial.
- Abstract semantics suffices for deductive soundness.
- If we require each language to have a formal abstract semantics, then there will be a problem of semantic regress.
- Hence we need foundations for abstract semantics, where the buck stops.
- Well-founded sets provide the most plausible universal foundation.
- This can also serve as the last resort on questions of consistency.
- Does not provide the best language in which to do mathematics.

A Semantic Stack

Illative Lambda Calculus / Type Theory
(more flexible, better structuring)

Non-well-founded ontology
(incorporating the full well-founded ontology)

Well-founded ontology

Well-Founded Set Theoretic Truth

Here is a concise and definite semantics for well-founded Set theory:

- a well-founded set is a definite collection of well-founded sets
- an interpretation of set theory is a transitive well-founded set
- a sentence is false if the collection of interpretations in which it is true is definite
- a sentence is true if the collection of interpretations in which it is false is definite

This semantics:

- is maximally rich (large cardinal axioms are true)
- is definite (CH has a truth value, particular facts about cardinal arithmetic have truth values)
- makes ZFC neither true nor false
- is not limited to first order languages (will work for infinitary set theory)
- is self defining (conjecture)

Consistency:

- consistent "monotone" sentences are true
- this generalises the idea that consistent large cardinal principles are true

Presentation slides and notes at:

www.rbjones.com