X-Logic

Roger Bishop Jones

April 18, 2008

Abstract

Architectural speculations concerning globally distributed automation of formal reasoning.

http://www.rbjones.com/rbjpub/isar/HOL/X-Logic.pdf
http://www.rbjones.com/rbjpub/isar/HOL/X-Logic.tgz

Id: root.tex,v 1.2 2006/11/13 07:12:37 rbj01 Exp

Contents

1 A Meta-Model

theory *XLMM2*
imports *Main*
begin

In theory XLMM2 [1] we model a kind of metalanguage for X-Logic. A model in isabelle contributing to X-Logic architecture and the design of XL-Glue. This version includes the use of signatures.

This second in a series of models in which the metatheory of X-Logic is developed, improves on the first model by allowing that programs read and write multiple documents, and by introducing the use of digital signatures to provide degrees of confidence in the truth of documents and the soundness of programs.

1.1 Abstract Syntax

The abstract syntax of our metalanguage is defined using a datatype in isabelle HOL.

The language is about documents (which are understood as propositions) expressed in various object languages, and programs (whose computations are interpreted as inferences) which read documents and create new documents. These documents languages and programs are all understood to inhabit the World Wide Web and each identified by a URI, which is a string. So we

[1] Id: XLMM2.thy,v 1.8 2007/06/11 20:48:54 rbj01 Exp

begin with a type abbreviation indicating that URIs are to be represented by strings.

types *URI* = *string*
  document = *URI*
  language = *URI*
  program = *URI*
  authority = *URI*

1.1.1 Sentences

The subject matter of the metalanguage is the truth of documents. The metalanguage permits the establishment (proof) of truth to be compounded from inferences performed by a variety of programs in various languages. From premises about the inferences performed by these various programs (which may be thought of as demonstrating lemmas) it is to be possible in the metalanguage to infer an overall conclusion.

The metalanguage therefore contains sentences which express the claim 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
4. a set of authorities are endorsed as infallible

datatype *sentence* =
    TrueDocs document list language list
| *ProgSpec program language list language list*
| *Compute program document list document list*

The significance of endorsements will become clearer shortly.

1.1.2 Judgements

In general sentences are not proven absolutely, but on the assurance of various authorities (sometimes called oracles). The combination of a sentence with a set of authorities which have contributed to our grounds for asserting the sentence is called a "judgement". For reasons connected with well-definedness of the semantics of judgements a judgement also contains a number. This may be thought of as a time-stamp, but is more loosely specified.

types *stamp* = *nat*

datatype *judgement* =
Assert stamp authority set sentence |
Endorse authority authority set

consts
jstamp :: *judgement* ⇒ *stamp*
jauths :: *judgement* ⇒ *authority set*
jsent :: *judgement* ⇒ *sentence*

primrec
jstamp (*Assert st as se*) = *st*
jstamp (*Endorse a as*) = *0*
primrec
jauths (*Assert st as se*) = *as*
jauths (*Endorse a as*) = {*a*}
primrec
jsent (*Assert st as se*) = *se*

1.2 Semantics

The set of authorities can be empty, but when asserted a judgement must be signed by an authority. The meaning of a judgement is that *if* all the authorities cited in the list have been hitherto infallible then the sentence is true.

However, the judgement is known only with that degree of confidence which we attach to the authority which asserts it (and has signed it), so even an unconditional judgement (one with an empty set of cited authorities) is still no better assured than its signing authority.

An authority has been "hitherto infallible" if all the judgements which it has signed with numbers less than that of the judgement in hand are true. In fallibility and truth are therefore mutually defined, the numbers attached to judgements relativise infallibility so as to make the mutual recursion well-founded.

The semantics of sentences and judgements is defined as truth valuations relative to appropriate interpretations.

1.2.1 Sentence Interpretations

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*

1.2.2 True Sentences

consts
truedoclist :: [*Sinterp*, *document list*, *language list*] ⇒ *bool*

primrec
truedoclist i (*h-d*#*t-d*) *ll* =
(*case ll of* [] => *False* |
(*h-l*#*t-l*) => (*docmap i h-d*) ∈ (*langmap i h-l*) & *truedoclist i t-d t-l*)
truedoclist i [] *ll* = (*case ll of* (*h-l*#*t-l*) => *False* | [] => *True*)

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*)

1.2.3 Judgement Interpretations

A judgement is true if infallibility of its authorities implies the truth of its sentence. To formalise this we need to talk about infallibility, and to talk about infallibility we need to have an interpretation which tells us which judgements have been affirmed by which authorities.

We therefore devise an extended interpretation for judgements in which a judgement map is available mapping each authority to the judgements it has affirmed.

types
judgement-map = *authority* ⇒ *judgement set*

record *Jinterp* =
sinterp::*Sinterp*
judgemap::*judgement-map*

1.2.4 Infallibility and Truth

Informally an authority is infallible if it only asserts true judgements. However, the definition of truth of a judgement will depend upon the infallibility of authorities, and this naive view does not lead to a well defined concept.

This is fixed by slighly *strengthening* the meaning of judgements, so that their truth depends only on the truth of *previous* judgements, and it is for this reason that judgements have been given a "stamp". This leads us to

the property of being "hitherto infallible" at some stamp value. This is the property that all judgements affirmed by the authority with smaller stamp values are true. It will be clear from the proof rules which we show later that this mechanism does not have to be implemented with timestamps.

One further complication is necessary, arising from endorsement. The infallibility of an authority is conditional on the infallibility of the authorities it has endorsed in a way which cannot be allowed for by attaching a truth value to the judgement in which the endorsement takes place. This is because the truth value of the endorsement can only depend on that of previous judgements, but the infallibility of an authority at some time depends on judgements made by authorities he has endorsed between the time at which the endorsement took place and the later time at which an infallibility judgement may be taking place.

Endorsements are therefore held to create a timeless partial ordering on authorities, and we require for the infallibility of an authority at some moment that neither he nor any greater authority has made a previous error. Greater in this case means directly or indirectly endorsed by the authority in question.

types

$inftest = authority \Rightarrow Jinterp \Rightarrow bool$
$truthtest = judgement \Rightarrow Jinterp \Rightarrow bool$

constdefs

$authr :: judgement\ set \Rightarrow (authority * authority)set$
$authr\ js == rtrancl\ \{p.\ \exists as.\ snd\ p \in as$
$\quad \& Endorse\ (fst\ p)\ as \in js\}$

$authrel :: judgement\text{-}map \Rightarrow (authority * authority)set$
$authrel\ jm == rtrancl\ \{p.\ \exists as.\ snd\ p \in as$
$\quad \& Endorse\ (fst\ p)\ as \in jm\ (fst\ p)\}$

$basett :: truthtest$
$basett\ j\ ji == case\ j\ of$
$\quad (Endorse\ a\ as) \Rightarrow True\ |$
$\quad (Assert\ n\ as\ s) \Rightarrow truesen\ (sinterp\ ji)\ s$

$itrec :: nat \Rightarrow (inftest * truthtest) \Rightarrow inftest$
$itrec\ n\ tsts\ auth\ ji == \forall a.\ (auth,a) \in authrel\ (judgemap\ ji)$
$\qquad \longrightarrow (\forall j.\ j \in judgemap\ ji\ a\ \&\ jstamp\ j < n \longrightarrow snd\ tsts\ j\ ji)$

$ttrec :: (inftest * truthtest) \Rightarrow truthtest$
$ttrec\ tsts\ j\ ji == (\forall auth.\ auth \in jauths\ j \longrightarrow (fst\ tsts)\ auth\ ji)$
$\qquad\qquad \longrightarrow basett\ j\ ji$

consts

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

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

constdefs
hitherto-infallible :: *nat* ⇒ *authority* ⇒ *Jinterp* ⇒ *bool*
hitherto-infallible n == *fst* (*ittt n*)

true-judgement :: *judgement* ⇒ *Jinterp* ⇒ *bool*
true-judgement j == *snd* (*ittt* (*jstamp j*)) *j*

1.3 Proof Rules

1.3.1 Inference

consts
thms :: *judgement set* ⇒ *judgement set*
syntax
yields :: *judgement set* ⇒ *judgement* ⇒ *bool* (**infix** ⊢ *25*)
translations
H ⊢ *p* == *p* : *thms*(*H*)

inductive *thms H*
intros

H: *p* : *H* ⟹ *p* : *thms H*

E: ⟦ *H* ⊢ *Assert n1* (*ll Un levels1*) *sent*;
H ⊢ *Endorse l ll*;
n1 < *n3*;
n2 < *n3* ⟧
⟹ *Assert n3* ({*l*} *Un levels2*) *sent* : *thms H*

TI: ⟦ *H* ⊢ *Assert n1 levels1* (*TrueDocs* [*d*] [*l*]);
H ⊢ *Assert n2 levels2* (*TrueDocs dl ll*);
n1 < *n3*;
n2 < *n3* ⟧
⟹ *Assert n3* (*levels1 Un levels2*) (*TrueDocs* (*d*#*dl*) (*l*#*ll*)):*thms H*

TEH: ⟦ *H* ⊢ *Assert n1 levels* (*TrueDocs* (*d*#*dl*) (*l*#*ll*));
n1 < *n2* ⟧
⟹ *Assert n2 levels* (*TrueDocs* [*d*] [*l*]):*thms H*

TET: ⟦ *H* ⊢ *Assert n1 levels* (*TrueDocs* (*d*#*dl*) (*l*#*ll*));
n3 < *n2* ⟧
⟹ *Assert n2 levels* (*TrueDocs dl ll*):*thms H*

MP: ⟦ *H* ⊢ *Assert n1 levels* (*TrueDocs idl ill*);
H ⊢ *Assert n2 levels* (*ProgSpec p ill oll*);

$H \vdash$ *Assert n3 levels* (*Compute p idl odl*);
n1 $<$ *n4*;
n2 $<$ *n4*;
n3 $<$ *n4* $\rrbracket$
$\Longrightarrow$ *Assert n4 levels* (*TrueDocs odl oll*):*thms H*

1.4 Properties

1.4.1 Soundness

A deductive system is a map from a set of judgments to the set of its consequences. It is sound if the consquences are true whenever the premises are.

constdefs
sound ::(*judgement set* $\Rightarrow$ *judgement set*) $\Rightarrow$ *bool*
sound f == $\forall$ *ji js*.
($\forall$ *j*. *j*:*js* $\longrightarrow$ *true-judgement j ji*) $\longrightarrow$ ($\forall$ *j*. *j*:(*f js*) $\longrightarrow$ *true-judgement j ji*)

1.4.2 Assurance

This property expresses the requirement that lowly assured premises do not affect highly assured conclusions, i.e. if you add more lowly assured judgements into your stock you do not increase the number of highly assured conclusions you can draw.

constdefs
filter ::*authority set* $\Rightarrow$ *judgement set* $\Rightarrow$ *judgement set*
filter as js == {*j* . *jauths j* $\subseteq$ *Image* (*authr js*) *as*}

assured ::(*judgement set* $\Rightarrow$ *judgement set*) $\Rightarrow$ *bool*
assured f == $\forall$ *js1 js2 as*. *filter as js1* = *filter as js2* $\longrightarrow$
filter as (*f js1*) = *filter as* (*f js2*)

end