Axiomatic Language

A Short Presentation

Walter W. Wilson
Lockheed Martin

Intl. Conf. On Logic Programming
Las Cruces, New Mexico
September, 2019

http://axiomaticlanguage.org/

Language Goals

1. pure specification – what, not how
2. minimal, but extensible
3. metalanguage – can subsume other languages

Specification by Enumeration

Idea: Program external behavior defined by infinite set of symbolic expressions that enumerate inputs and corresponding outputs.

Recipe

- pure, definite Prolog with Lisp syntax
- higher-order generalization [HiLog]
- string variables

The Core Language (1)

Axioms generate valid expressions.

expression:

atom, – `` `abc ``, `` `+ ``

expression variable, – `%x`, `%1`

sequence of >=0 expressions and **string variables**

– ``(`M () $1 %)``

The Core Language (1)

Axioms generate valid expressions.

expression:

atom, – `` `abc ``, `` `+ ``

expression variable, – `%x`, `%1`

sequence of >=0 expressions and **string variables**

– ``(`M () $1 %)``

axiom: **conclusion** expr and >= 0 **condition** exprs

<*conclu*> < <*cond1*>, …, <*condn*>.

<*conclu*>. ! unconditional axiom

The Core Language (2)

axiom instance: substitute values for variables in axiom

```
    (`A %x $1)< (`B %x), (`C $1).
→ (`A `x `u ())< (`B `x), (`C `u ()).
```

The Core Language (2)

axiom instance: substitute values for variables in axiom

```
  (`A %x $1)< (`B %x), (`C $1).
→ (`A `x `u ())< (`B `x), (`C `u ()).
```

valid expressions: If all conditions of an axiom instance are valid expressions, the conclusion is valid.

```
  (`a `b).
  ((%) $ $)< (% $).

 → (`a `b),
   ((`a) `b `b),
   (((`a)) `b `b `b `b),
          …
```

Example – Natural Numbers

Set of natural numbers:

```
(`num (`z)).
(`num (`s $))< (`num ($)).
   → (`num (`s `s `z))
```

Addition of natural numbers:

```
(`plus %n (`z) %n)< (`num %n).
(`plus %1 (`s $2) (`s $3))<
  (`plus %1 ($2) ($3)).
  → (`plus (`s `z) (`s `z) (`s `s `z))
```

Syntax Extensions

single char in single quotes:

```
'A' = (`char (`0 `1 `0 `0  `0 `0 `0 `1))
```

char string in single quotes within sequence:

```
(… 'abc' …) = (… 'a' 'b' 'c' …)
```

char string in double quotes:

```
"abc" = ('abc') = ('a' 'b' 'c')
```

symbol not starting with special char:

```
abc = (` "abc")
```

Example – List Predicates

Concatenation of sequences:

```
(concat ($1) ($2) ($1 $2)).
  → (concat (a b) (c d e) (a b c d e))
```

Member of a sequence:

```
(member % ($1 % $2)).
  → (member b (a b c))
```

Reverse of a sequence:

```
(reverse () ()).
(reverse (% $seq) ($rev %))<
  (reverse ($seq) ($rev)).
  → (reverse (u v) (v u))
```

Conclusion

- Specifications – software engineering benefit
  - Smaller, more readable, more reusable, more correct
- Minimal & pure – well-suited to proof
  - Equivalence of specification and program
  - Prove assertions to validate specification
- Billion-dollar application!
  - http://axiomaticlanguage.org/A_Vision_for_CAD_released.pdf
- Implementation grand challenge
  - Transformation of specifcations to programs
  - http://axiomaticlanguage.org/LOPSTR18_LM_released.pdf