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
Axioms generate valid expressions.

**expression:**

- **atom:** \( \sim \text{abc}, \sim + \)
- **expression variable:** \( \%x, \%1 \)
- **sequence of \( \geq 0 \) expressions and string variables**

\[- (\sim \text{M} ( ) \sim $1 \% )\]
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
axiom instance: substitute values for variables in axiom

\((\forall x \; 1) < (\forall y), (\forall z)\).
\rightarrow (\forall x \; x \; u () < (\forall y \; x), (\forall z \; 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 (\010\000\000\001)')

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:

\[
\text{(concat \(\$1\) \(\$2\) \(\$1 \$2\))}.
\]
\[
\rightarrow \text{(concat \(a \ b\) \(c \ d \ e\) \(a \ b \ c \ d \ e\))}
\]

Member of a sequence:

\[
\text{(member \% \(\$1 \% \$2\))}.
\]
\[
\rightarrow \text{(member \(b\) \(\text{\(a \ b \ c\)}\))}
\]

Reverse of a sequence:

\[
\text{(reverse \() \()\)}.
\]
\[
\text{(reverse \(\% \$\text{seq}\) \($\text{rev} \%$\))}<
\]
\[
\text{(reverse \(\$\text{seq}\) \($\text{rev}$\))}.
\]
\[
\rightarrow \text{(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!
- Implementation grand challenge
  - Transformation of specifications to programs