# Proof, Implementation, and CAD Application Challenges of Axiomatic Language

Walter W. Wilson
wwwilson@acm.org
Lockheed Martin, Fort Worth, Texas

Intl. Conf. on Logic Programming 2020

http://axiomaticlanguage.org/ICLP20RC.pdf
http://axiomaticlanguage.org/ICLP20RC_slides.pdf
http://axiomaticlanguage.org/ICLP20RC_video.mp4

# Language Goals

1. pure specification – what, not how
2. minimal, but extensible
3. metalanguage – easy to define new language features

# 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

# An Example

a set of **axioms**:

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

generated **valid expressions**:

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

# Example – List Predicates

```
(concat ($1) ($2) ($1 $2)).    ! concatenation
  -> (concat (a b) (c d e) (a b c d e))

(member % ($1 % $2)).          ! member predicate
  -> (member c (a b c d))

(reverse () ()).               ! reverse function
(reverse (% $seq) ($rev %))<
  (reverse ($seq) ($rev)).
  -> (reverse (u v) (v u))
```

# Natural Numbers and their Addition

```
(num 0).                  ! natural numbers
(num (s %n))< (num %n).
  -> (num 0), (num (s 0)), (num (s (s 0))), …

(plus %n 0 %n)< (num %n).     ! addition
(plus %1 (s %2) (s %3))< (plus %1 %2 %3).
  -> (plus (s 0) (s 0) (s (s 0)))   -- 1 + 1 = 2

(== % %).         ! identical expressions
```

# Proof in Axiomatic Language

a possible axiom:

```
(num (s (s %)))< (num %).  ! 2+n is num if n is num
```

– no additional valid expressions – a **valid clause**

commutativity of addition:

```
(== %3a %3b)< (plus %1 %2 %3a), (plus %2 %1 %3b).
```

see http://www.axiomaticlanguage.org/proof.htm

# Implementation of Axiomatic Language

Map specification to efficient program

1. “understand” the input specification – match axioms against a knowledge base of programming concepts
2. generate efficient program from this understanding using pre-stored algorithm knowledge

Goal: Automatic transformation of straightforward specifications for most typical problems, else expert must add knowledge.

http://axiomaticlanguage.org/BabySteps.pdf

# Application to Computer Aided Design

http://www.axiomaticlanguage.org/A_Vision_for_CAD_released.pdf

- Represent CAD data in a declarative language

  -- instead of a vendor's proprietary binary file format
- High-level definitions capture engineering knowledge
- Programmability supports design automation & optimization
- Open-source geometric engine – accessible mathematics
- Long-term accessibility of design data

  http://www.axiomaticlanguage.org/LOTAR_Thoughts.html

Axiomatic language is ideal host for this embedded DSL

- Elegant long-term standard for data preservation
- Can prove geometric algorithms correct

# Conclusions

- Specifications – smaller, more readable, more reusable, more correct
- Minimal and pure - well-suited to proof
- CAD – a billion-dollar application!
- “Extended axiomatic language” – a form of negation

  http://www.axiomaticlanguage.org/EAL.html
- Axiomatic language needs a home at a university
- These research challenges may need the energy of youth!