My Favorite Verbose Programming Technique

Aaron W. Hsu aaron@dyalog.com, Dyalog, Ltd.

LambdaConf 2025, Estes Park, CO

I have an extreme preference
for brutally simple, concise code.

“You can’t do X with Y.”

I can, I will.

Needs to be practical.

Needs to be a good experience,
result in better code.

My compiler became an answer to

“You can’t do tree manipulation naturally in APL.”

APL can pretty much directly express solutions
to all the problems addressed by traditional computer science

Except…

Event-driven Reactive Behaviors

What makes event-driven problems a bad fit?

Non-deterministic event arrival

Potential need to respond “immediately”

Open Question for a Long Time

I want a “good” solution!

What makes a good solution?

Inexpensive/Low-overhead

Low Abstraction

High Level

Domain-centric

Concise

Mathematical

Easy proofs of correctness

“Pen and Paper” friendly

Why those requirements?

APL as a specification language

User Requirements

→

Specification

→

Implementation

→

Testing

User Requirements are human language expressions of behavior

Specifications are formal definitions of user requirements

Programs are typically an implementation of an (implied) specification

Formal Specifications cannot be Mechanically Verified

A human is the only source of
“the right thing”

Specification

→

Implementation

Specification

→ [Types]

Haskell/Scala/Scheme/JavaScript

Specification

→

APL

Specification

+

Implementation

APL = Specification + Implementation

Where do existing solutions fail?

Functional Reactive Programming

Callback Handlers

Event Loop over switch(event)

Where do existing solutions fail?

Assume the presence of an understood, correct specification

Rely on ad hoc assurance of completeness

Still depend heavily on human verification

Types have all these same issues

What am I afraid of?

“Have I forgotten anything?”

“Have I done anything silly?”

“Are we sure this is the right thing?”

Time to go looking for inspiration…

I’m in APL, it’s a math notation,
what do the mathematicians do?

Basically, state machines.

What prior art exists around this sort of thing in Iverson-land?

J has a built-in state machine operator!

Documentation explicitly discusses this problem.

That’s two votes for state machines.

But they still have these issues:

Are we correct?
Did we miss anything?
Did we do something silly?

Enter Cleanroom Software Engineering…

Famous for making developers prove their code,
without being allowed to run it.

“Write bug-free code.”

Cleanroom also taught a method of functional specification as state machines!

The emphasis was:

Correct – Am I doing the right thing?

Complete – Am I missing something?

Consistent – Am I doing something silly?

Sequence-based Enumeration

A specific **process** for modeling system behavior
by building a correct, consistent, and complete
state machine based on **human-language requirements**

System behavior

≡

f(seq-hist, stimulus) → response

f(seq-hist, stimulus) → response

Systematically define f by:

Defining the sets of stimuli and responses

Begin systematically enumerating all possible sequence histories

For each history, consider

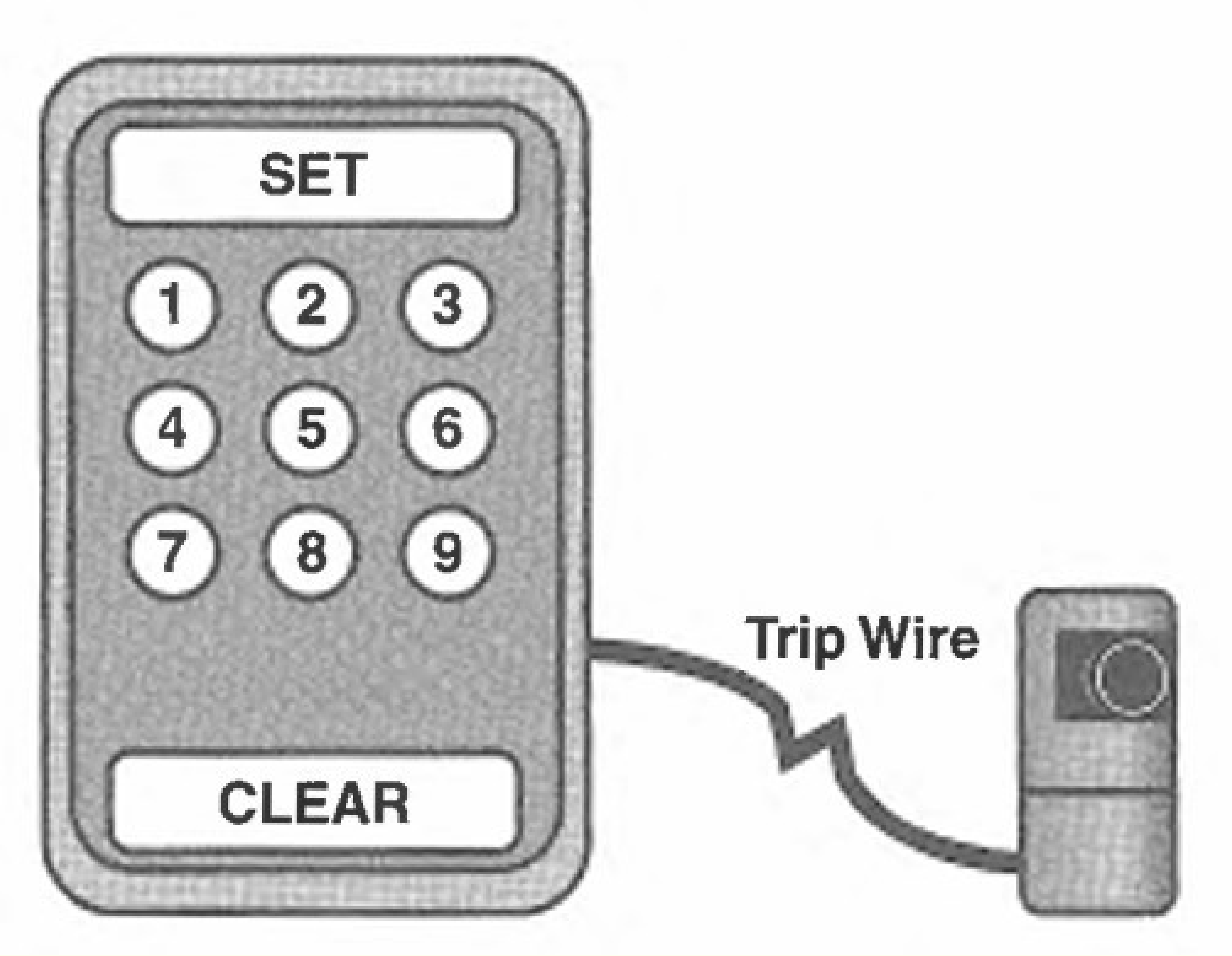
for each stimuli:

What is the response?

What is the “equivalent sequence?”

If no equivalent, then this is a canonical sequence
→
new state in the machine.

Example from Cleanroom Book: Security Alarm


SET
1
2
3
4
5
6
7
8
9
CLEAR
Trip Wire

Process → Specification evolution:

Stimuli Abstraction

Response Refinement

The key emphasis:

The systematic, human, exploratory process
over the entire state space.

The primary benefit

You must confront every possible state
and make an explicit decision about your system

Pretty much eliminates your ability to have unexpected behavior

Model checkers and the like can prove nice temporal properties

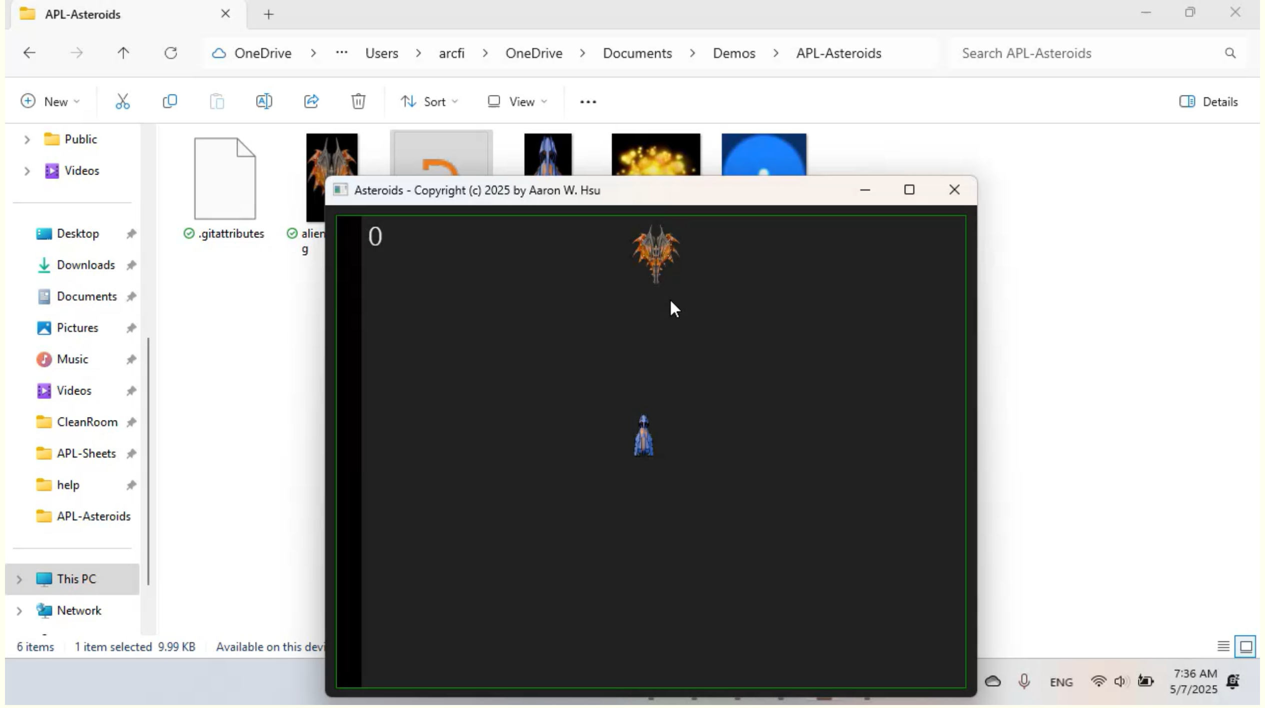
Secondary Benefits

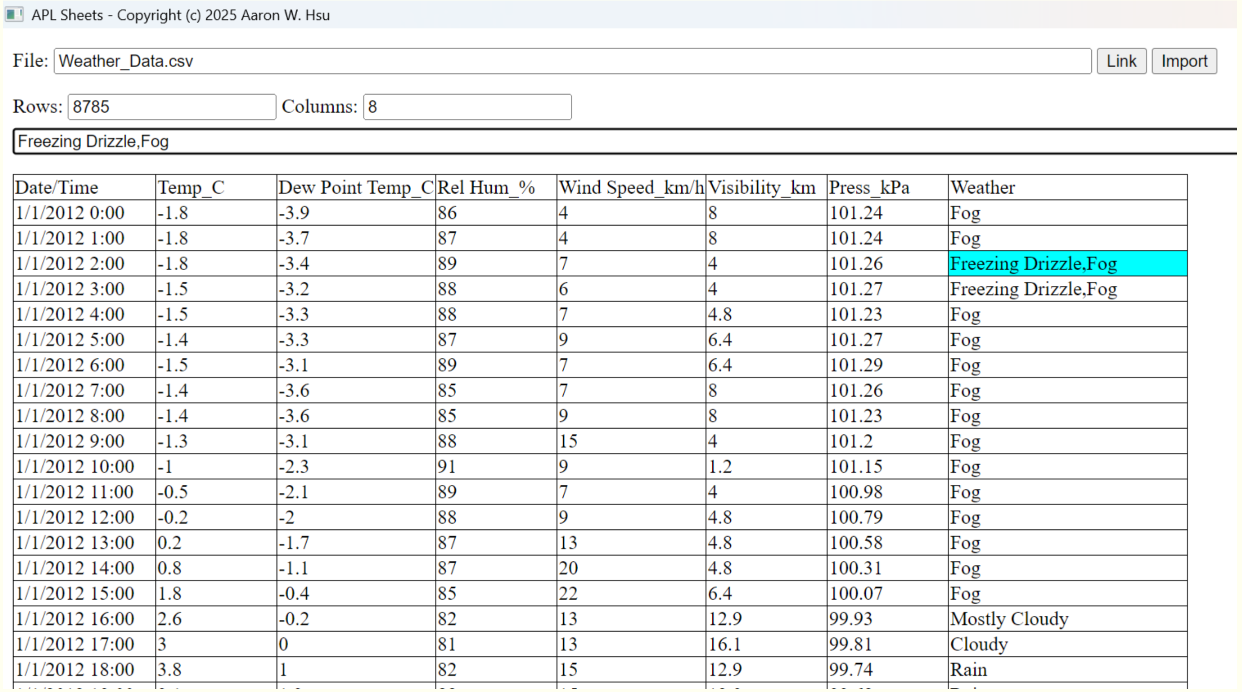
Very low overhead

Low abstraction

Flexible and general

[It’s just computed goto!]





File: Weather_Data.csv

Rows: 8785 Columns: 8

Freezing Drizzle,Fog

Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
1/1/2012 0:00	-1.8	-3.9	86	4	8	101.24	Fog
1/1/2012 1:00	-1.8	-3.7	87	4	8	101.24	Fog
1/1/2012 2:00	-1.8	-3.4	89	7	4	101.26	Freezing Drizzle,Fog
1/1/2012 3:00	-1.5	-3.2	88	6	4	101.27	Freezing Drizzle,Fog
1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog
1/1/2012 5:00	-1.4	-3.3	87	9	6.4	101.27	Fog
1/1/2012 6:00	-1.5	-3.1	89	7	6.4	101.29	Fog
1/1/2012 7:00	-1.4	-3.6	85	7	8	101.26	Fog
1/1/2012 8:00	-1.4	-3.6	85	9	8	101.23	Fog
1/1/2012 9:00	-1.3	-3.1	88	15	4	101.2	Fog
1/1/2012 10:00	-1	-2.3	91	9	1.2	101.15	Fog
1/1/2012 11:00	-0.5	-2.1	89	7	4	100.98	Fog
1/1/2012 12:00	-0.2	-2	88	9	4.8	100.79	Fog
1/1/2012 13:00	0.2	-1.7	87	13	4.8	100.58	Fog
1/1/2012 14:00	0.8	-1.1	87	20	4.8	100.31	Fog
1/1/2012 15:00	1.8	-0.4	85	22	6.4	100.07	Fog
1/1/2012 16:00	2.6	-0.2	82	13	12.9	99.93	Mostly Cloudy
1/1/2012 17:00	3	0	81	13	16.1	99.81	Cloudy
1/1/2012 18:00	3.8	1	82	15	12.9	99.74	Rain

Conclusion

arcfide@sacrideo.us

Sequence-based enumeration can be used as a foundation for defining an executable specification of event-driven behavior.

https://www.sacrideo.us/last-minute-discount-for-apl-workshop/