
Computational Thinking for Lawyers

Kevin D. Ashley

Professor of Law and Intelligent Systems

University of Pittsburgh

ashley@pitt.edu

Computational Thinking for Lawyers

1. Debugging the (legal) code by
 - a) Resolving logical ambiguities (i.e., normalization)
 - b) Evaluating proposed rule with test cases
 - e.g., real and hypothetical counterexamples, exceptions
2. Algorithms, flowcharts, process models
 - ❑ to explain complex legal structures and processes
 - ❑ e.g., debugging proposed rule with test case,
 - ❑ statutory interpretation, predicting outcomes, structured arguments
3. Computer tools for conceptual/logical queries
 - ❑ not Boolean queries with key words but
 - ❑ conceptual hypotheses to test against data;
 - ❑ need legal IR/AI systems that process conceptual queries intelligently

1. Debugging legal rules by: a) normalizing

- Syntactic ambiguity almost always present and unintentional.
- Technical issue: "scope" of a logical operator
- For example:
 - No person may send an unsolicited email from within WA State to an address the sender knows is held by a WA State resident and that contains a commercial message and contains a misrepresentation in the subject line or transmission path.
- Is an email in violation if it contains a misrepresentation but no commercial message??
 - Not in violation if:

AND

**NOT commercial
NOT misrepresentation**

v.

NOT

**AND
commercial
misrepresentation**

1. Debugging legal rule: b) with test case

■ Proposed rule:

- IF unsolicited email sent from within WA State to WA State resident AND
- Contains commercial message OR contains misrepresentation
- THEN violation.

■ Suppose:

- **Test Case:** X sends unsolicited commercial email from Spokane to Seattle and message goes via Portland, OR.

■ Rule too broad

- regulates interstate commerce
- contrary to another rule, Commerce Clause of U.S. Constitution
- Federal law trumps state law

2. Algorithms, flowcharts, process models:

e.g., Debugging with Test Case*

→ 1. Propose rule for deciding current fact situation (cfs):

- ❑ Construct a rule that leads to a favorable decision in the cfs and
- ❑ is consistent with applicable underlying legal principles/policies and important past cases,
- ❑ and give reasons.

← 2. Pose test case to probe if rule is too *broad*:

- ❑ Find/construct a test case that:
- ❑ emphasizes some normatively relevant aspect of the cfs and
- ❑ to which the proposed rule applies and assigns the same result as to the cfs, but
- ❑ where, given legal principles/policies, the result in test case is normatively wrong.

→ 3. Respond to test case:

(3.a) Justify the proposed rule:

- ❑ Analogize the test case and the cfs and
- ❑ argue that they both should have the result assigned by the proposed rule. *Or*

(3.b) Modify the proposed rule:

- ❑ Distinguish the test case from the cfs, argue that they should have different results and that the proposed rule yields the right result in the cfs, and
- ❑ add a condition or limit a concept definition so that the narrowed rule still applies to the cfs but does not apply to, or leads to a different result for, the test case. *Or*

(3.c) Abandon the proposed rule and return to 1.

*Ashley, Lynch, Pinkwart, Alevan, 2008

3. Computer tools to support conceptual/logical queries

- Given process model
 - (e.g., debugging with test case)
- If need a test case
 - (e.g., to probe rule as too broad)
- search for example that:
 - emphasizes some normatively relevant aspect of the cfs and
 - to which proposed rule applies and assigns same result as to cfs, but
 - where, given legal principles/policies, that result is normatively wrong in the example.
- i.e., case where
 - state anti-spam statute:
 - held invalid due to supervening rule, or
 - that deals with intrastate addresses held invalid, or
 - held invalid under Commerce Clause

Caution!

- Legal problem solving is highly context-dependent in ways that may not be anticipated.
- Be cautious about recommending computational thinking to law students
 - in case it leads to them to focus more on a mechanical application of a pre-defined method rather than on
 - the context of and opportunities in the problem to-be-solved.

2. Algorithms, flowcharts, process models:

e.g., statutory interpretation*

- A. In interpreting statutory provision, consider 3 types of argument:
1. linguistic arguments
 2. systemic arguments
 3. teleological-evaluative arguments.
- B. Accept as *prima facie* justified a clear interpretation
- i. at level 1 unless there is some reason to proceed to level 2;
 - ii. where level 2 invoked for sufficient reason, accept as *prima facie* a clear interpretation at level 2 unless there is reason to move to level 3.
 - iii. if at level 3, accept as justified only the interpretation best supported by the whole range of applicable arguments.
- C. Take account of arguments from intention and other transcategorical arguments (if any)
- as grounds which may be relevant for departing from the above *prima facie* ordering.

* MacCormick, D. and Summers, R. Eds. 1991. *Interpreting Statutes: A Comparative Study*. P. 531. Aldershot: Dartmouth.

2. Algorithms, flowcharts, process models: e.g., case-based legal prediction*

Input: Current fact situation

1. Identify legal issues
2. For each issue determine favored party:
 - If factors favor same side, return side, else
 - Perform evidential reasoning with cases:
 - If cases found with issue-related factors
 - Test hypothesis that majority side should win
 - Explain-away counterexamples
 - Otherwise, Broaden-Query
3. Combine analysis from issues

Output: Predicted outcome and explanation