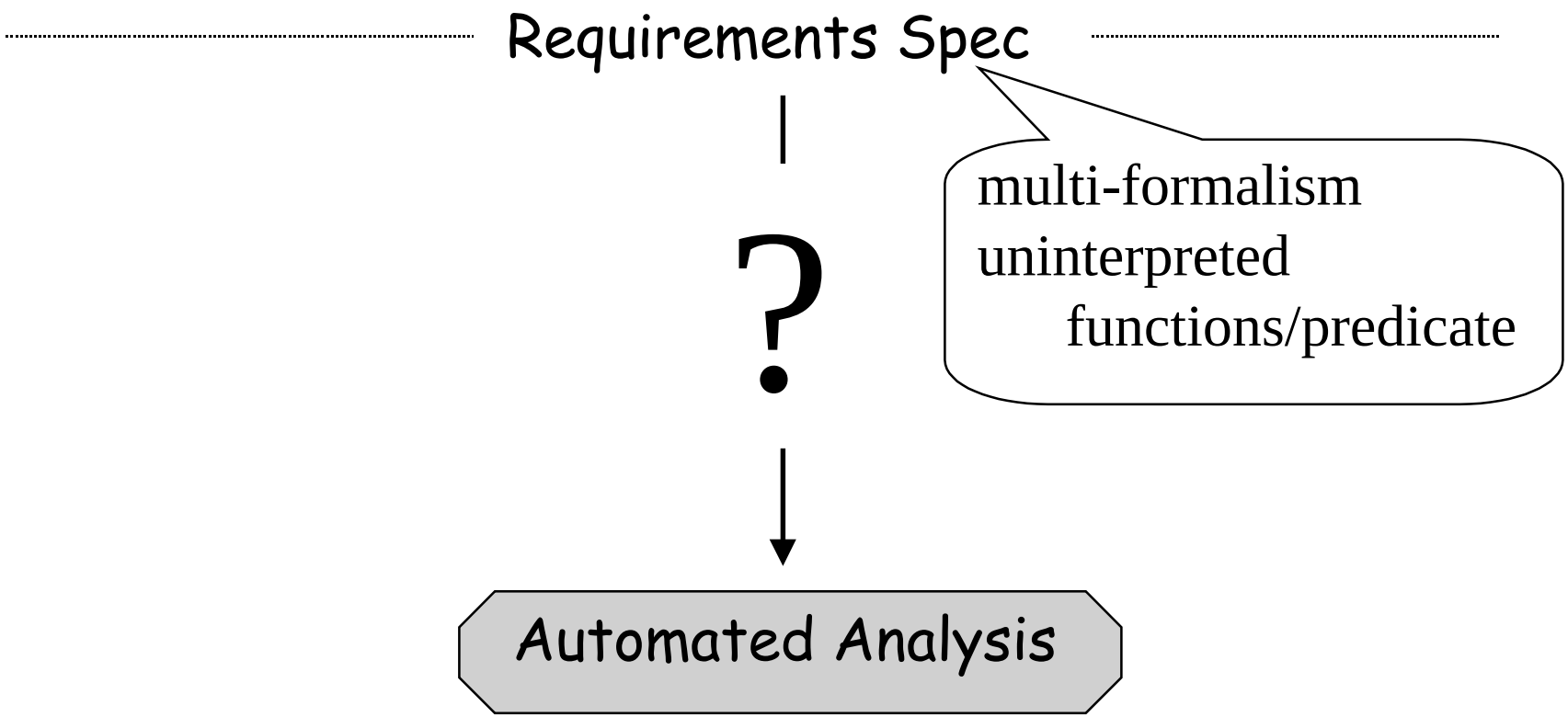
Formalization and Analysis of the Separation Minima for Aircraft in NAT

Nancy Day
University of British Columbia

Jeff Joyce, Gerry Pelletier
Hughes Aircraft of Canada

day@cs.ubc.ca
http://www.cs.ubc.ca/spider/day

Thesis: Formal Validation of System Specifications

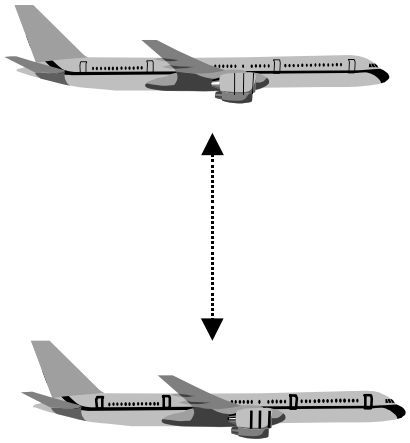


Outline

- system
- formal description technique
- analysis techniques
- analysis results
- future work
- summary

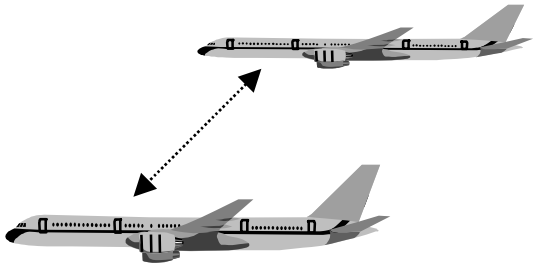
Separation Minima for NAT

rules for air traffic controllers



Vertical Separation

Lateral Separation



Longitudinal Separation

Existing Specification

- complex decision logic
- stateless

Form of the specification:

- informal natural language (64 pgs)
  - thoroughly reviewed and in use
- pseudocode interpretation (9 pgs)

Analysis

- completeness
  - are all combinations of inputs for two flights covered ?
  - what are the cases covered by the default values ?
- consistency
  - is it ever possible that the specification has multiple possible outcomes for the same inputs ?

Goals

- both specification and analysis results had to be readable and reviewable by the domain expert
- formal specification:
  - in a suitable notation
  - don't add details
- analysis - return results:
  - in the terms given in the specification

Formal Specification

- combination of:
  - ASCII based predicate logic
    - » defined types, functions and predicates types
    - » uninterpreted types, functions and predicates
  - function and predicate tables
    - » slight variation of AND/OR tables
- 15 tables; 16 defns; 47 uninterpreted functions and predicates
- 18 pg document; formal spec = 300 lines

Example

:flight;
FlightLevel : flight -> num;
IsLevel: flight -> bool;
:typeOfAircraft := Turbojet | Supersonic | Other;
TypeOfAircraft : flight -> typeOfAircraft;

OR →

AND ↓

			Default
FlightLevel(A)	_ < 280	_ > 450	
TypeOfAircraft(B)	_ = Turbojet	_ = Supersonic	
IsLevel(A)	_ = T	.	
InCruiseClimb(A)	.	_ = F	
Separation (A,B)	**1000**	**4000**	**2000**

structure captures related elements in a row

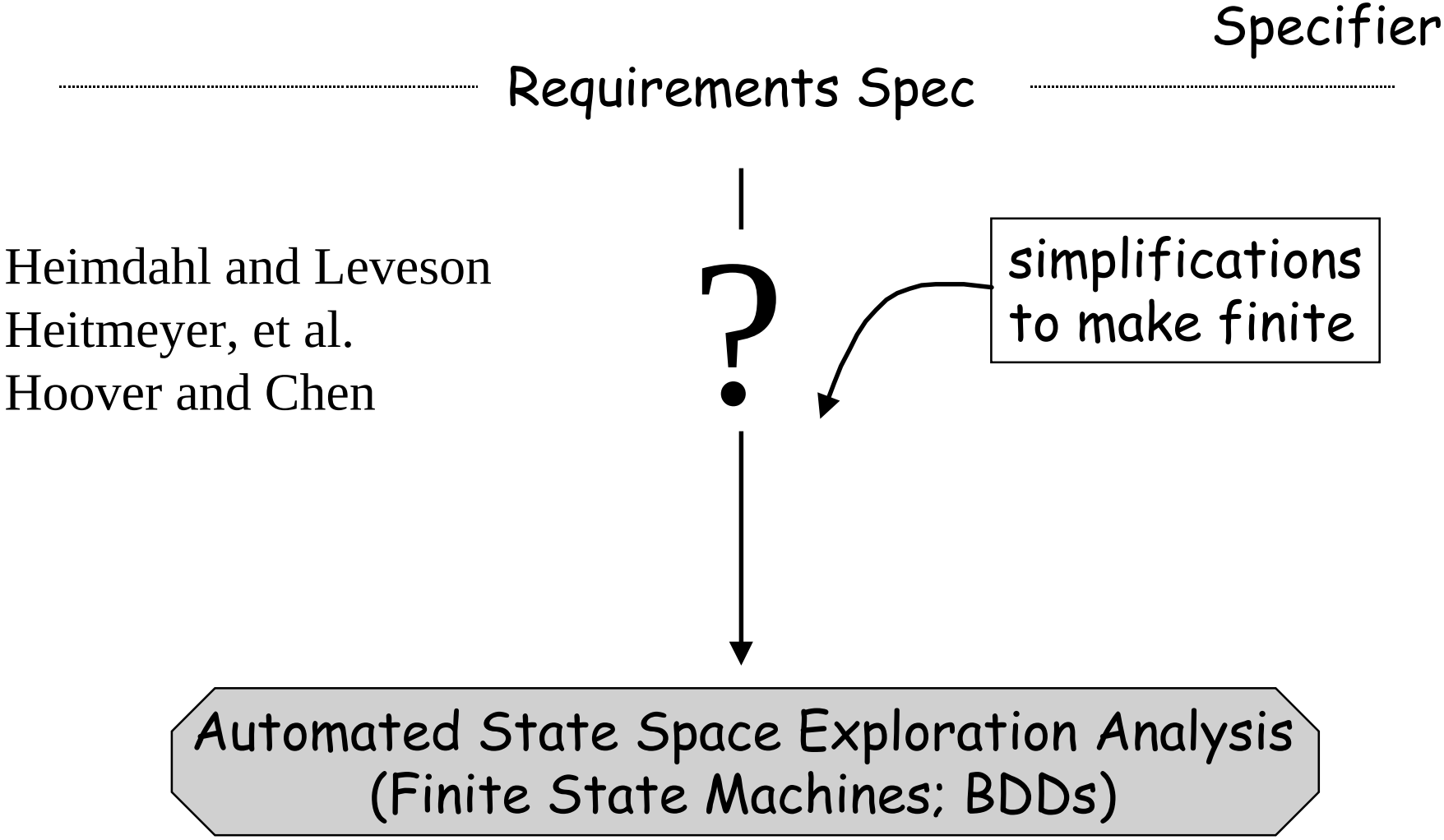
Definitions of Completeness and Consistency of a Table

- completeness:
  - default cases
- consistency:
  - no two columns with differing result values for the function overlap
- with respect to possible values for row entries

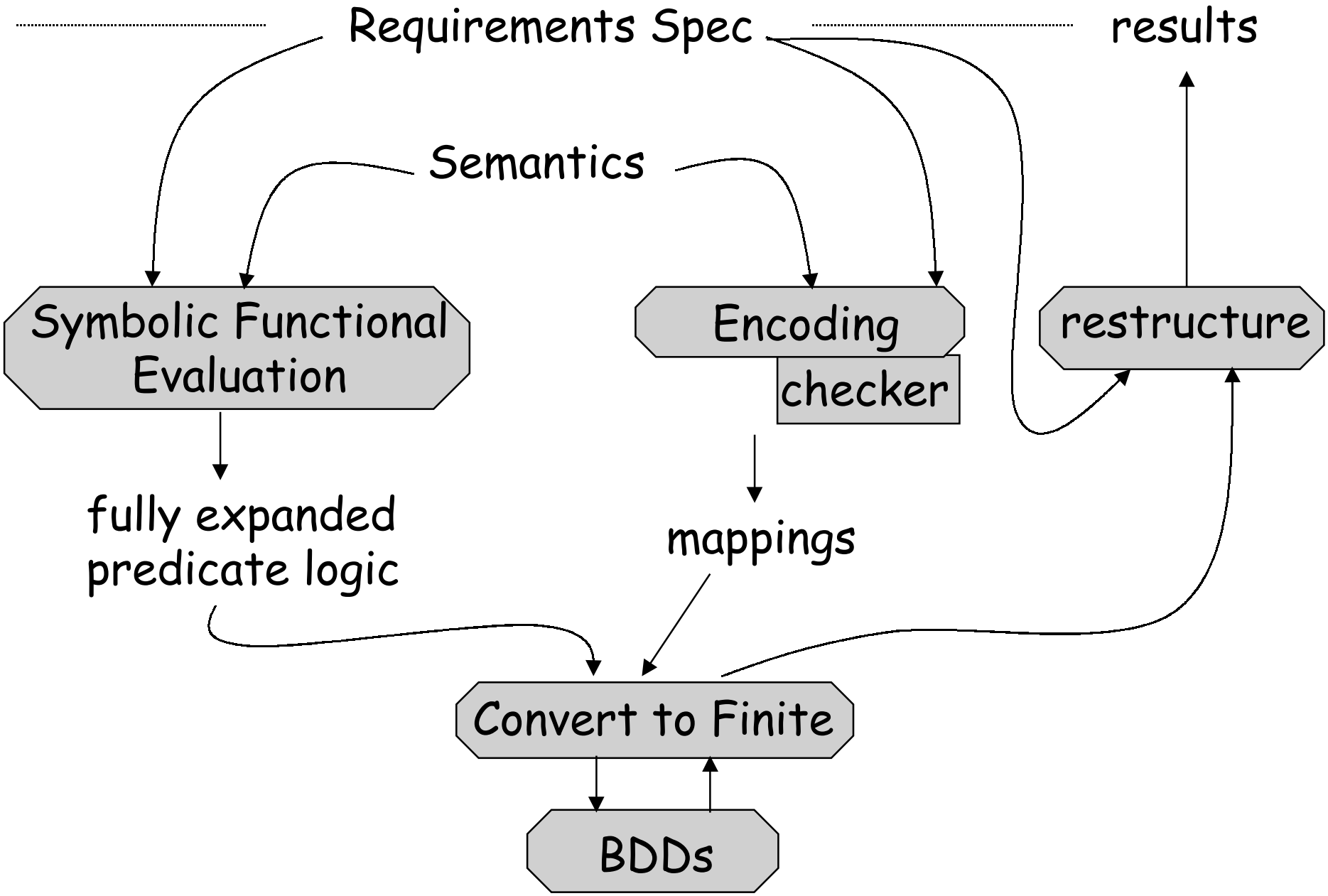
Outline

- system
- formal description technique
- analysis technique
- analysis results
- future work
- summary

Analysis :Context



Framework


Requirements Spec
results
Semantics
Symbolic Functional Evaluation
Encoding
checker
restructure
fully expanded predicate logic
mappings
Convert to Finite
BDDs

Key features of this approach

- direct use of explicit defn of semantics
  - integrate notations
- symbolic functional evaluation
- encodings based on structure
- reverse mappings and structuring of results

Analysis Results

- completeness analysis found:
  - missing assumptions "everyone knew about" (domain knowledge)
  - incorrect partitions
- consistency analysis found:
  - three places where the requirements are inconsistent
- symmetry analysis found:
  - assumptions about the uninterpreted terms

Environmental Assumptions

- predicate logic constraints

forall A:flight. NOT (IsLevel(A) AND InCruiseClimb(A));

- analysis:
  - applied to existing items of the correct type
  - existentially quantified out of result
- limited output
- allowed symmetry checking to be more successful

Future Work

- extending framework to do model checking of specifications with state, such as statecharts
- aeronautical telecommunications network (ATN)

Summary

- analysis technique:
  - handles uninterpreted functions/predicates
  - definition of the semantics used in analysis
    - » integrate multiple notations
  - takes advantage of structure found in tables
  - user can add environmental assumptions
  - return results in terms and structure of specification

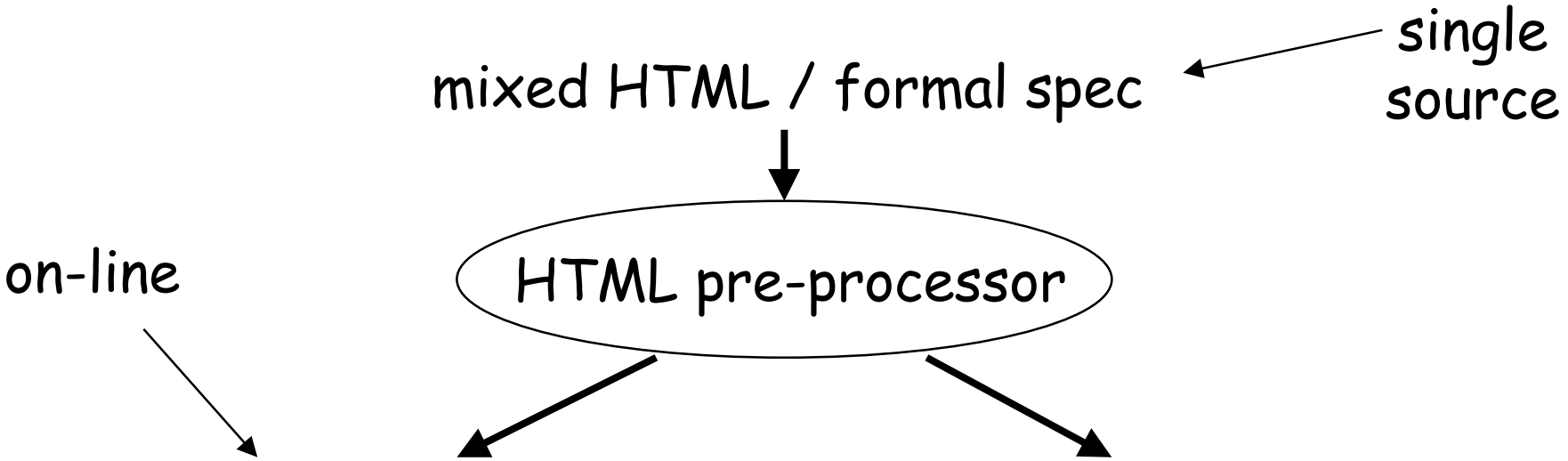
extend what could be done fully automatically

Thesis: Validation of System Specifications

- multi-formalism specifications
- uninterpreted functions/predicates
- exploit structure in specifications
- let user control the analysis through their knowledge of the domain

extend range of what can be automatically analyzed

Presentation of the Specification



That's all!

Example of Analysis Results

Advantages / Contributions

- use the explicit defn of semantics directly in analysis; also simulation, prototyping; analysis of semantics
- general framework for:
  - multiple notations; multiple analysis techniques
  - non-formal methods person; formal methods expert
- return results at correct level of abstraction
- exploit inherent abstractions

Related Work

- builds on existing work by
  - Heimdahl and Leveson
  - Heitmeyer,
  - especially for definitions of compl/cons
- PVS
  - similar approach but
    - » didn't have to add a construct to the language
    - » compl/cons not required by defn of semantics
    - » enumerating resultant cases
    - » took advantage of environment

Embedding

- had to get into a common form for analysis
- S: ASCII notation based on higher order logic; open tool support
- predicate logic parts were given directly in S
- tables were embedded in S using a textual representation

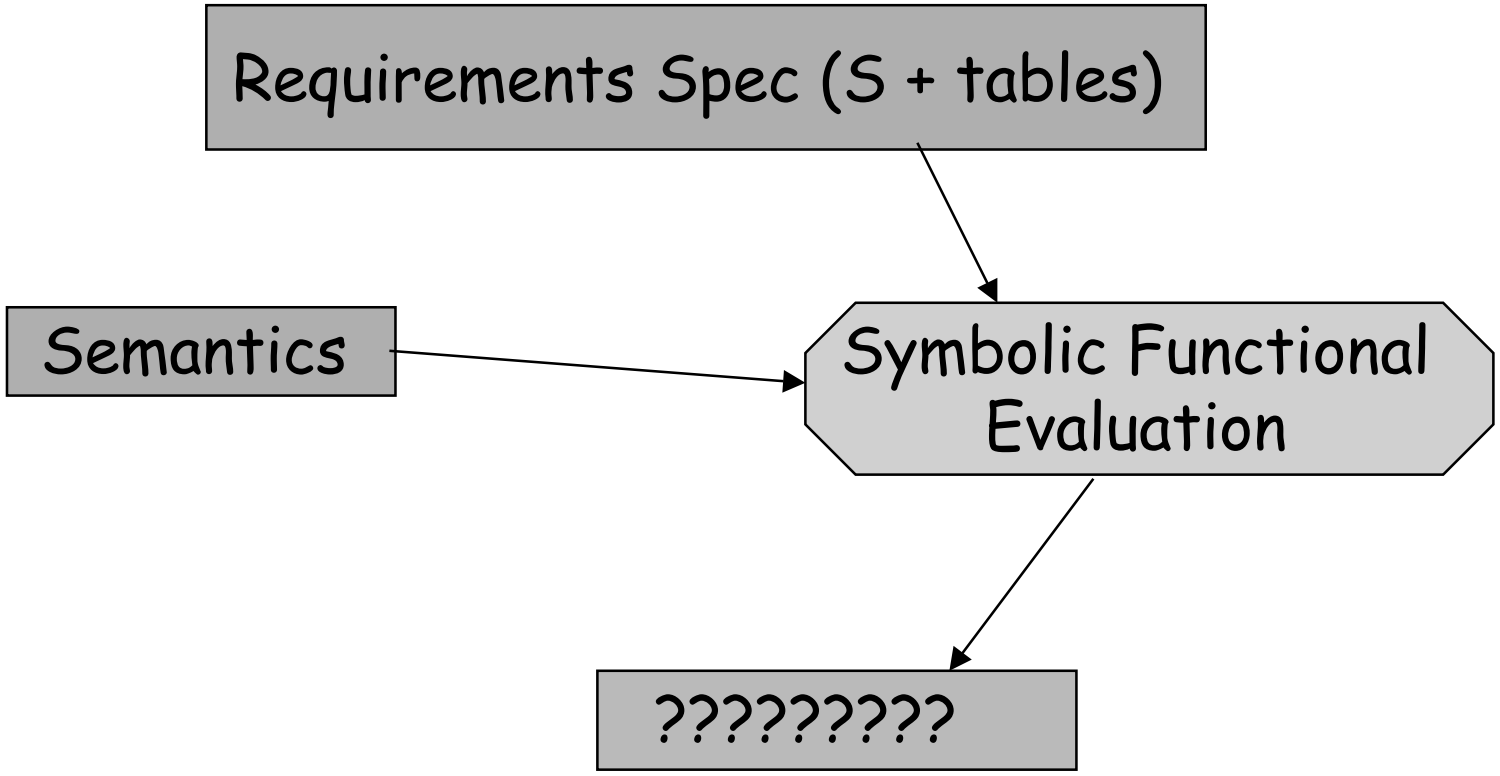
Semantics of the tables

- also written in S (given in Appendix)
- keyword "Table" or "Predicate Table" is the semantic function
- "executable" in that they can reduce the table into something in terms of the entries in the rows and Boolean connectives

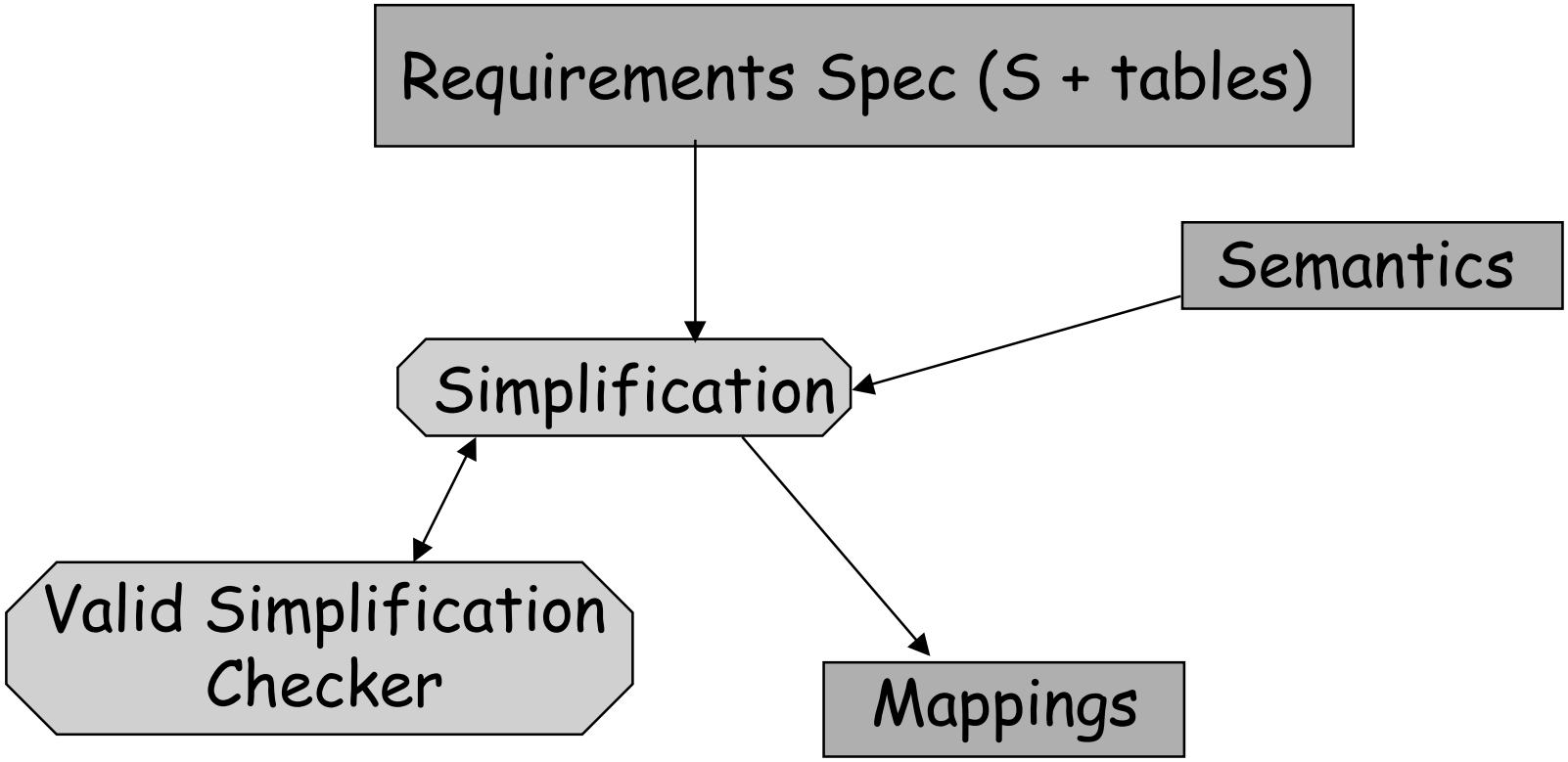
Thesis Statement

Having an explicit machine-readable operational semantics for a notation within a common framework provides a systematic way to exploit inherent abstractions to carry out state-space exploration analysis.

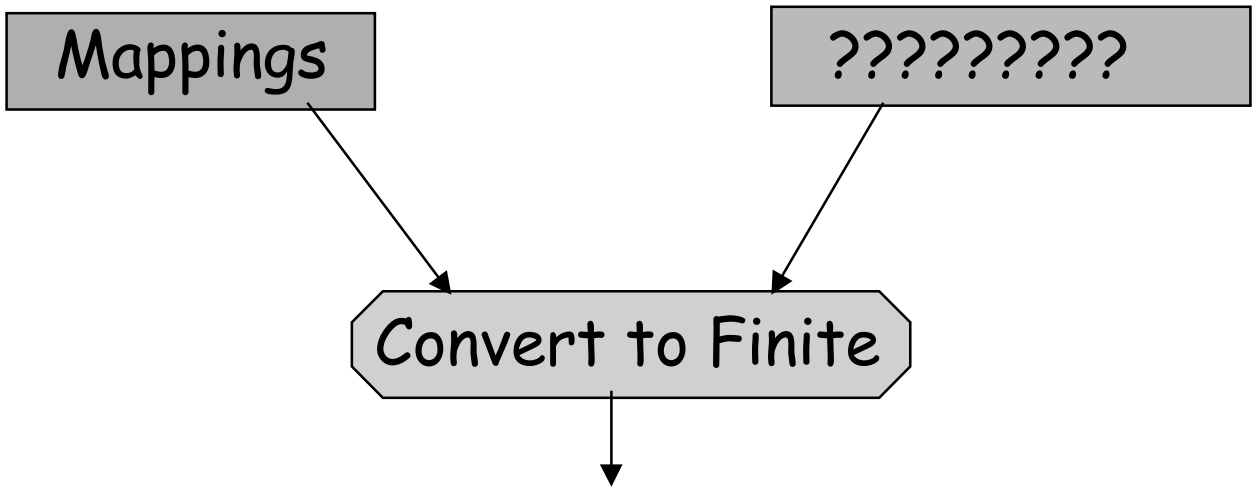
First Step: Dealing with Multiple Notations



Second Step: Determining Valid Abstractions

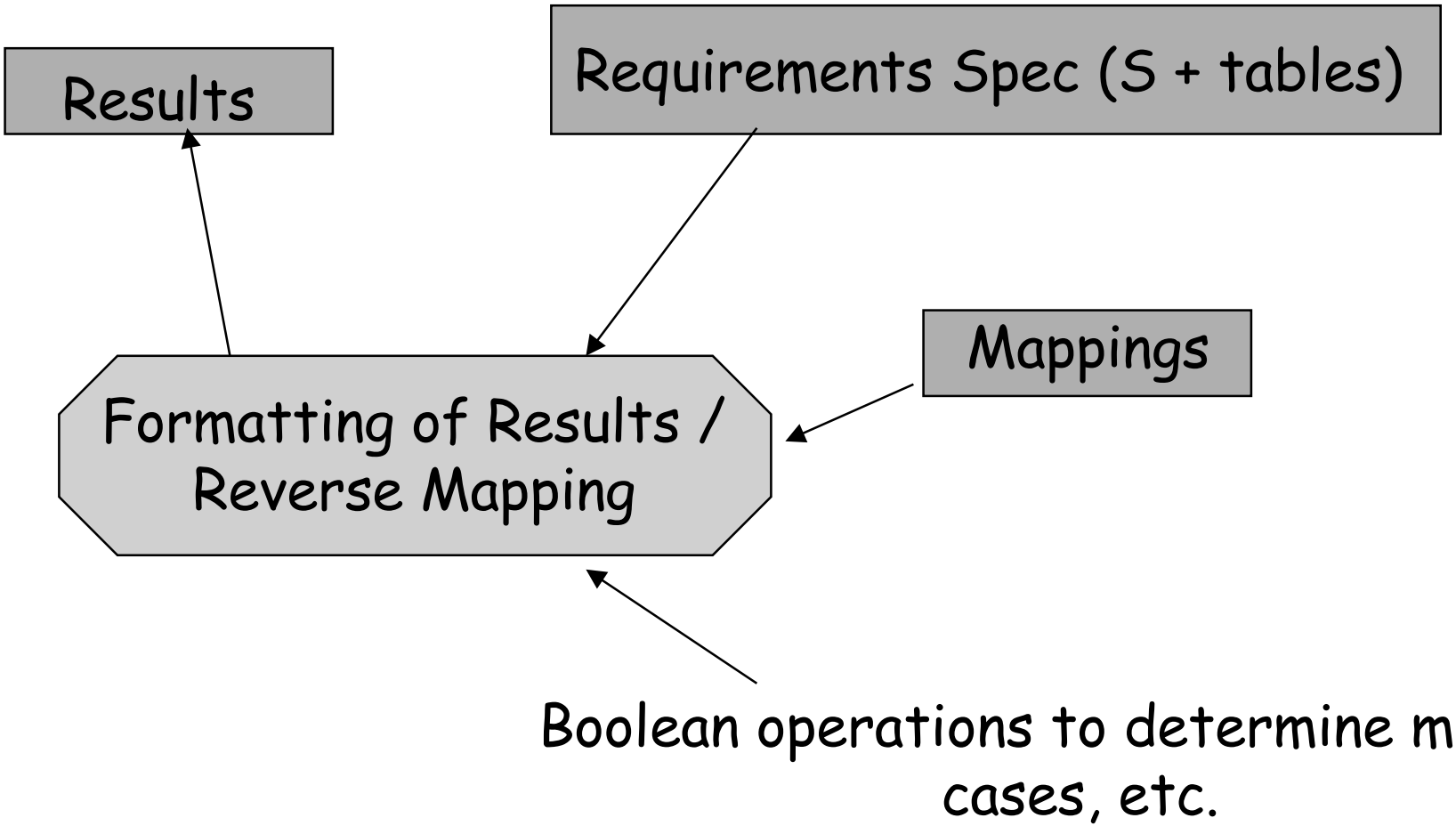


Third Step: Mapping to a Finite Domain and ???



Boolean operations to determine missing cases, etc.

Fourth Step: Returning Results



fix this picture