Overview

- **Motivation**
  - Limited Domain relations (LD-relations)
  - What happens for out of domain arguments?
  - Difficulty parsing complex mathematical specifications
  - What does that do, again?

- **Objective**
  - Clear, formal representation of requirements specifications in a tabular format
  - Mathematical foundation
  - Possible for both human and machine parsing
  - Better partitioned specifications

- **Techniques Used**
  - Modified predicate logic
  - Provided formal description of tabular relations

- **Origin**
  - A-7 aircraft specifications

Techniques

- **Predicate Logic**
  - Logic Definition
    - Provides intro to naive set theory
  - Mathematical presentation of components of an expression
  - Requirements
    - Either true or false (no "maybe")
    - Simple notation
    - Avoid nesting
    - Avoid explicit quantification
  - Solution: Out of domain arguments result in entire expression evaluating to false

- **Tabular Representations**
  - Mathematical presentation of tables
    - Defined for some whole number of dimensions
    - Usually only two dimensions used
  - Tables allow for partitioning of the domain, behavior, state, and range of an expression
  - Range from simple single variable domain and range specifications to vector relations
Example

- A program that searches an array for a value $x$

<table>
<thead>
<tr>
<th>$y_i$</th>
<th>$(y_i &gt; x)$</th>
<th>$(y_i &lt; x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Impact

- **Predicate Logic**
  - Began the dissection of mathematical notations
  - Impact of LD-relations
  - Foundation for mathematical based logics with computational context

- **Tabular Representations**
  - Provided first specification of tabular notation
  - Formed the foundation of a number of requirements based specification languages
    - SCR
    - RSML
    - SpecTRM-RL

- **Applications**
  - A-7 specification, robots, nuclear power plants, flight management, etc.

Shortest Path Algorithm Formal Program Documentation

Doan K. Pehur
Parnas' Display Method

- Peters demonstrates program documentation using the display method
- Display method
  - Presented in a different Parnas paper
  - A method for program documentation
  - A program is presented as a set of displays that can be independently verified
  - Displays use Parnas' tabular notation
- Peters discusses display method
  - Aspects of describing C program/procedure specifications
  - Demonstrates use in specifying a procedure from the greatest common denominator algorithm
  - Demonstrates use in specifying Dijkstra's shortest path algorithm

A Procedure in Greatest Common Denominator

- Compares 2 ints and replaces largest with their difference
  - Watch out for aliasing!

```
void gcd(int *x, int *y)
{
    int x = *x;
    int y = *y;
    if (x > y) {
        *x = x - y;
    } else {
        *y = y - x;
    }
}
```

Shortest Path Algorithm

- Shortest path algorithm is broken up into displays for programs and subprograms
  - Subprogram displays can be checked independently
- Still substantially complex
  - But easier to verify
- Demonstrates a practical application of display method
Automated Consistency Checking of Requirements Specifications

C. L. Heitmeyer, R. D. Jeffords, and B. G. Lehne
ACM Transactions on Software Engineering and Methodology
vol. 5, pp. 231-261, July 1996

SCR (Software Cost Reduction) method

- SCR tables are derived from Parnas' tables
- Motivation
  - SCR tabular specifications describe systems concisely and unambiguously
  - Analysis takes many man-hours
  - Still prone to human error
  - Need tools to automate creation and analysis
- Objectives
  - An automated "consistency checker" will check various system independent properties of specifications
    - Would be more accurate
    - Would allow more human effort to be used in analyzing less trivial aspects of specifications
  - A prototype consistency checker is tested to analyze the specification of the A-7 Operational Flight Program

Example SCR Tables

- Event table: variables are set by events
  - Overridden as a function of Pressure, Block, Reset

<table>
<thead>
<tr>
<th>Mode</th>
<th>Event</th>
<th>Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Block</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Reset</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

- Condition table: variable values depend on others
  - SafetyInjection as a function of Pressure, Overridden

<table>
<thead>
<tr>
<th>Mode</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Overridden</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Safety Injection</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>NGT Override</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
Consistency Checker Details

- Some properties checked:
  □ Proper syntax
  □ Type correctness of variables
  □ All variables must be initialized
  □ Reachability (all states reachable from initial configuration)
  □ Disjointness (specifications must be deterministic)
  □ Coverage (entire range of conditions must be covered)
  □ Lack of circularity among dependencies

- Results of prototype test on A-7 spec
  □ The specification had been rigorously checked by two independent teams for errors
  □ In a running time of 245 seconds, the automated checker found:
    □ 17 legitimate errors over 11 condition tables
    □ 57 legitimate errors over 3 large state transition tables

---

Designing Specification Languages for Process Control Systems: Lessons Learned and Steps to the Future

Nancy G. Lesman, Mats P. E. Hedinhall, and Jon Damon Reese
ESAC/SIGSOFT/IPSJ, pp. 127-145, 1999

---

SpecTRM-RL

- Motivation
  □ Lessons learned from experience with RSML
  □ Complicated mathematical specs result in ambiguous specifications
    that facilitate errors in implementation
  □ Traditional formal specifications are difficult to read and interpret
    □ Specs are inaccessible to those not familiar with the specification language
    □ Based on experience with FAA personnel involved with the
      TCAS II project

- Objective
  Provide a formal specification of software requirements that combines
  state machines and tabular representations that makes the meaning
  more accessible to non-computer professionals
SpecTRM-RL

- **Technique**
  - "Make the specification user-centered rather than easy to analyze or faithful to mathematical conventions."
  - Two primary concepts
    - Semantic distance: how closely the representation of the model matches the users mental model of the process
    - Distinct separation of requirements and design
  - Tables only show what and why, not how

- **State Based with Tables**
  - The conditions for state transitions in non-trivial systems are very complex
  - Propositional logic statements quickly become unwieldy
  - **Solution**: Use AND/OR tables to represent sets conditions and behaviors
  - AND/OR tables used to detail the specification of the transitions for a given state diagram

---

SpecTRM-RL Example

![SpecTRM-RL Example](image)

- **INITIAL** — Unknown
- **Unknown, Not-initialized** — Not initialized
- **Compress CLamp** — open for No AA
- **Door Closed** — open for AA
- **Compress CLamp** — close for AA
- **Door Closed** — open for AA

---

Requirements Specification for Process-Control Systems

Nancy G. Leveson, Mats P. E. Höökreh, Holly Hildreth, and Jon D. Rose
Requirements Specification Modeling Language (RSML)

- **Relationship to SpecTRM-RL**
  - RSML is the predecessor of SpecTRM-RL
  - Both share the goal of making specifications more accessible to experts in the field the application is being developed for

- **Motivation**
  - The primary motivation of RSML was the FAA's request for a more readable specification for the TCAS II system.

- **RSML**
  - Much more state diagram based than SpecTRM-RL

- **What's the difference?**
  - A number of usability changes were made from RSML to SpecTRM-RL
    - No more internal broadcast events
    - More focus on state transitions represented as AND/OR tables
  - Leveson et. Al cite Parnas in SpecTRM-RL paper but not in the RSML paper...

### Motivation

The primary motivation of RSML was the FAA's request for a more readable specification for the TCAS II system. RSML is the predecessor of SpecTRM-RL, both sharing the goal of making specifications more accessible to experts in the field the application is being developed for. RSML is much more state diagram based than SpecTRM-RL.

**What's the difference?**

A number of usability changes were made from RSML to SpecTRM-RL:

- No more internal broadcast events
- More focus on state transitions represented as AND/OR tables

**Leveson et. Al cite Parnas in SpecTRM-RL paper but not in the RSML paper...**

### Conclusions

- The overall goal was to make requirements specifications more readable by partitioning the domain, range, state, and behaviors of some portion of a specification into more palatable partitions.

- **Realizations of this goal:**
  - Predicate logic simplifies LD-relations
  - Parnas' tables used to partition specification expressions
  - SCR tables
  - AND/OR tables similarly partition logical expressions

- **End Result:** More readable requirements specifications that unambiguously handle partial functions and take advantage of two dimension representations of complex expressions.