3-1-2005

Supporting Documentation for the 2003 SPS-Prospec Experiment

Oscar Mondragon

Salamah Salamah

Comments:
UTEP-CS-05-06.

Technical Report UTEP-CS-05-06
Supporting Documentation for the 2003 SPS-Prospec Experiment

Oscar Mondragon[1] and Salamah Salamah[2]
[1]European Software Institute
oscar@esi.mx
[2]University of Texas at El Paso
salamah@cs.utep.edu

1 Introduction

In spring of 2003 an empirical study was conducted to compare the effectiveness of the *Prospec tool* [1] with the Specification Pattern System (SPS) [2]. The objective of the experiment was to determine the effect that Prospec and SPS have over the completeness and correctness of the generated software property specifications.

The purpose of this document is to present the material that was used during the experiment, and to document how the classification of the patterns and scopes of each trial was validated.

2 Experiment Methodology

The *experiment subjects* were undergraduate students, graduate students, researchers, and instructors from computer science departments at three institutions: University of Pennsylvania (UPenn), the University of Texas Pan American (UTPA), and the University of Texas at El Paso (UTEP). The subjects self reported varying degrees of knowledge and experience about requirements engineering. The subjects lacked experience with both SPS and Prospec; however, subjects were given a 30-minute tutorial about: a) patterns and scopes; b) specification of properties using patterns and scopes; c) identification of incomplete property descriptions, e.g., defining relations among propositions; and d) navigation of Prospec tool and SPS web site. The tutorial can be

found in Appendix C. The *experimental objects* in this experiment were English descriptions of software properties. Table A1 summarizes the experimental objects, which were assigned to Set A and Set B. Each set has five properties with different types of patterns, e.g., *universality*, *existence*, and *response*. The experimental subjects were not aware of the distribution of types of patterns within the set of properties. Table A2 presents the proper pattern and scope combination for each property along with any composite proposition of presentation of pattern or scope limit.

There are two sets of properties, named A and B. Each set has five properties, one for each type of pattern: universality, absence, existence, precedence, and response. The subjects were not aware of the distribution within the set of properties. Sets A and B have properties at the same level of complexity.

Table 1: Experiment layout showing allocation of property set and tools to each group.

Group	Part I		Part II	
	Tool	**Property set**	**Tool**	**Property set**
G1	Prospec	A	SPS	B
G2	SPS	B	Prospec	A
G3	Prospec	B	SPS	A
G4	SPS	A	Prospec	B

The study was conducted as follows:

1) Subjects read and signed a consent form (Appendix B)
2) Subjects were handed an instructions form (Appendix B)
3) Subjects filled out the *Prospec Pre-Evaluation* form (Appendix B) prior to the study.

4) Subjects were equally distributed and randomly assigned to four groups, named G1, G2, G3, and G4, as outlined in Table 1. The process for such assignment was as follows:
    a) The number of subjects for each session was a multiple of four.
    b) The administrator labeled pieces of papers as G1, G2, G3, and G4, respectively. There was a total of (number of subjects/4) pieces of paper for each group.
    c) The administrator placed the papers into a bag.
    d) Subjects retrieved a paper from the bag for their group assignment.
5) Subjects attended a 30-minute tutorial on the types of patterns and scopes, Specification Pattern System (SPS), and the Prospec tool.
6) Groups G1 and G3 specified properties using first Prospec and then SPS. Subjects in G1 first specified properties from set A using Prospec and then properties from set B using SPS. Subjects in G3 first specified properties from set B using Prospec and then properties from set A using SPS.
7) Groups G2 and G4 first specified properties using SPS and then Prospec. Subjects in G2 first specified properties from set B using SPS and then properties from set A using Prospec. Subjects in G4 first specified properties from set A using SPS and then properties from set B using Prospec.
8) The propositions were randomly assigned to sets A and B based on each type of pattern using a random number generator for the first assignment of the first type of pattern.

9) The order of the properties within sets A and B was equally distributed and randomly assigned for each subject. The order was assigned using a random number generator, but the order in which the subjects specified properties was not to be enforced.
10) For each specified property, the subject filled out either a *Prospec Property Log* or *SPS Property Log* (Appendix D), depending on the tool being used.
    a) The property log was pre-marked with the Group ID and Property ID. Group ID was G1, G2, G3, or G4. Property ID was coded as follows: <Set-name><Property-number>, where Set-name was either 'A' or 'B,' and Property-number was an integer from 1 to 5.
    b) Subjects were assigned a number. The Subject ID was coded as follows: <Institution-number><Number>. The following assignments were made for each institution: 0 – Pilot; 1 – UTEP; 2 – UTPA; 3- U Penn. Number is an integer value. Group and subject assignment logs can be found in Appendix E.
11) Subjects filled out the *Prospec Post-Evaluation* and *SPS Post-Evaluation* forms (Appendix F) depending on the tool being used.

3 Verification of Trials

Once the survey was completed, the subjects responses were summed up in the matrices shown in Appendix H. These matrices represent the subjects' choice of a pattern/scope combination for each of the properties A1 through B5. In some cases the total number of responses among properties and tools varied because subjects did not always register a response.

The correct choice is indicated by a bold letter **(C)** in the appropriate cell. The correct choice was determined by thorough examination of the properties. In particular,

each choice was tested against the test cases given in Appendix G. In all cases, no counter examples were found. Appendix I gives the counter example for a response that was selected by a significant number of subjects and that was deemed incorrect. Patterns and scope that used the same propositions were ignored since the subjects were instructed not to do this. For example, property A1 clearly has only two propositions; “all elements are removed from the list” and “the list is deleted”. However, six subjects chose the Precedence/Before-R combination of pattern and scope which means that there needs to be three propositions defined; T and P for the pattern and R for the scope. This means that a proposition has to be used more than once. Another example is Property A2 in which 10 subjects chose the Precedence/After-L combination. However, all of those subjects either failed to specify propositions for pattern and scope or used the same proposition twice. After examining the property logs for these properties, the property was ignored.

In the case of property B3, seven subjects choose the Universality/After-L combination rather than Absence/After-L. This was considered as a correct choice since the subjects negated the proposition P, and Universality(~P) is equivalent to Absence(P).

4 References

[1] Mondragon, O. and A. Q. Gates,”Supporting Elicitation and Specification of Software Properties through Patterns and Composite Propositions,” Intl. Journal Software Engineering and Knowledge Engineering, 14(1), Feb. 2004.

[2] Specification Patterns, http://patterns.projects.cis.ksu.edu/, September 2004.

Appendix A
Survey objects

Table A1: English descriptions of software properties used as survey objects

SET	NO	NAME	PROPERTY DESCRIPTION
A	1	LIST	A list shall remove all its elements before it can be deleted.
	2	BRAKING	When a command expires, the train shall stop propulsion, reconfigure braking, and apply maximum braking.
	3	AVIONICS	Transition to mode 602 or mode 605 shall not occur until the entry maneuver is calculated.
	4	WHEEL	If wheel slippage is detected when maximum braking is applied, the braking force shall be reduced in order to regain wheel adhesion.
	5	TRAIN	From the time that maximum braking is applied to the time when the train actually stops, a train shall not enter a closed gate and shall keep a safe distance from the train in front.
B	1	PERFORMANCE	When system performance is degraded, the scheduler shall do one of the following: a) Swap blocked processes to the hard drive, b) Increase the CPU time slice assigned to processes, c) Stop spawning processes.
	2	ATM	Within a withdraw transaction in the ATM system; the user account shall be updated before the money is given, the card is returned, and the receipt is printed.
	3	BIOS	Once the BIOS has been loaded and the system has been initialized, the OS shall not install new devices, i.e., mouse and joystick.
	4	PROCE	A process shall release all its locks before it terminates.
	5	PAGING	If a system has less than 2MB and more than 1 MB of free memory, the memory management daemon shall execute the paging process.

Table A2: Pattern and scope for the properties in Set A and Set B (see A3 for meaning of propositions)

Prop	Scope	CP	Pattern	CP
A1	After L until R L = ¬LE, R = LE	N/A	Absence P P = LD	N/A
A2	Global	N/A	Response(P, T) P = CE; see CP for T	$Eventual_c$ T_1 = SP, T_2 = RB, T_3 = MB
A3	Global	N/A	Precedence(P, T) P = CEM; see CP for T	$AtLeastOne_c$ T_1 = M602 , T_2 = M605
A4	After L until R See CP for L, R = ¬WS	$Parallel_c$ L_1 = WS, L_2 = MB	Universality (P) P = RB	N/A
A5	After L Until R L = MB, R = TS	N/A	Universality(P) See CP for P	$Parallel_c$ P1 = ¬CG, P2 = SD
B1	Global	N/A	Response(P, T) P = PD; see CP for T	$AtLastOne_c$ T1= SP, T_2 = TS, T_3 = SS
B2	After L Until R L = WT, R = ¬WT	N/A	Precedence(P, T) P = AU; see CP for T	$AtLeastOne_c$ T_1 = MG, T_2 = CR, T_3 = RP
B3	After L See CP for L	$Eventual_c$ L_1 = LB, L_2 = IS	Absence P P = NS	N/A
B4	After L Until R L = ¬ RL, R = RL	N/A	Absence P P = PT	N/A
B5	After L Until R L = 2MB, R = ¬1MB free memory	N/A	Universality P P = PR	N/A

Table A3: Meaning of propositions used in Set A and Set B.

Property	Proposition and meaning
A1	LE = List is empty, LD = List is deleted
A2	CE = Command has expired, SP= Train stops propulsion, RB = Braking is reconfigured, MB = Maximum braking is applied
A3	CEM = Entry maneuver is calculated, M602 = Transition to mode 602 occurs, M605 = Transition to mode 605 occurs
A4	WS = Wheel slippage is detected, RB = Braking is reduced, MB = Maximum braking is applied
A5	MB = Maximum braking is applied, CG = Train is not in closed gate, TS = Train is stopped, SD = Train is within safe distance
B1	PD = System performance is degraded, SP = Block processes are swapped, TS = CPU time is increased, SS = Spawning processes are stopped.
B2	WT = Withdraw transaction is in process, AU = Account is updated, MG = Money is given, CR = Card is returned, RP = Receipt is printed
B3	LB = BIOS is loaded, NS = OS has installed new device, IS = System is initialized
B4	RL = Process has released all locks, PT = Process has terminated
B5	2MB = “Free system memory < 2 MB” , 1MB = “Free system memory > 1 MB, PR = Paging is running

Appendix B
Pre-expirement Forms

Form B-1: Consent form

CONSENT FORM

Prospec is a tool that assists practitioners in the elicitation and specification of system properties. Practitioners are guided by directed questions, definitions, and graphics. The tool provides visual and textual guidance for specifying common properties of concurrent systems and facilitates:

- identification of incomplete descriptions of software properties with respect to missing relations among multiple propositions,
- definition of sequential and concurrent behavior,
- identification and specification of patterns and scopes,
- reduction of errors in the formal specification.

The study will determine:

- The level of support that the tool provides (state diagrams, timeline diagrams, Petri nets, decision trees, and symbols) to identify the pattern, scope, and relations among propositions.
- Whether the specification ordering for scope or pattern is important.
- The number of errors in the specification of a property done by using the tool and manually.

The participant will receive a 30-minute tutorial in property specification using the Specification Pattern System (SPS) and Prospec. The participant will be required to specify properties with the use of Prospec and without the use of Prospec. The results will be used to determine the level of assistance that is provided by Prospec.

Only the name of participants is recorded. No data, i.e., scores and responses, will be identified in study records by individual names of participants. Data will be number coded for confidentiality. The study does not impose any risk to the participants. There will be no cost to the participant. The study can be completed in two sessions, approximately two hours each.

Please contact Oscar Mondragon (915-747-8012, Hoscar@cs.utep.eduH) or Dr. Ann Q. Gates (915-747-6413, Hagates@cs.utep.eduH) if you have any questions. You may also contact the office of ORSP 915-747-7939 to report a research concern.

Please read and sign this consent form. Your participation in this study is completely voluntary and you may withdraw at anytime. Your involvement will help us evaluate and refine Prospec tool. Thank you for your time and help.

I agree to participate and I understand that all the information that I provide will remain confidential at all times. My participation is voluntary.

_____ _____

Participant name and signature Date

Form B-2: Expirement Instructions

Instructions

1. You have been assigned to one of the following groups: G1, G2, G3, G4.
2. If you are in Group G1 or G3, you will specify five properties using Prospec first.
    a. You must complete the Property Log form for each property.
    b. Print the property specification after completing the specification (see instructions below)
    c. Turn in the Property Log forms for each property and the Prospec Post-Evaluation form before starting the SPS specifications.
    d. Note the time that it took to complete the first set of properties and mark it at the top of the Post-Evaluation form.
    e. Specify the next five properties using SPS. Complete and turn in the corresponding forms.
    f. Record the time for the second set of properties.
3. If you are in Group G2 or G4, you will specify five properties using SPS first.
    a. You must complete the Property Log form for each property.
    b. Turn in the Property Log form for each property and the SPS Post-Evaluation form before starting the Prospec specifications.
    c. Note the time that it took to complete the first set of properties and mark it at the top of the Post-Evaluation form.
    d. Specify the next five properties using Prospec.
    e. Print the property specification after completing the specification (see instructions below)
    f. Complete and turn in the corresponding forms.
    g. Record the time for the second set of properties.

Printing Properties in Prospec

To save a property:
Select “File” from the main menu, then click on the “save” option.

To print a property:

1. Select a property from the “Properties window.”
2. Select “Report On” from the main menu, and then click “Selected property.”
3. Prospec displays a message indicating the name of the file where the property was saved. (The name of the file is the name of the selected property with extension .txt.)
4. Open Microsoft explorer and print the file.

Form B-3: Pre-Expirement Evaluation of Subjects

Pre-Evaluation Form

Subject ID:	Group ID:

5 DEMOGRAPHICS

Age group: Under 20 ( ) 20 – 35 ( ) Over 35 - 50 ( ) Over 50 ( )

Gender: Male ( ) Female ( )

Status:

( ) Student Classification (FR, SO, JR, SR, GR): _____

( ) Faculty

( ) Professional Occupation _____

Years of experience in software development: _____

SKILLS INVENTORY

Please rate your level of experience by checking the appropriate box, where

- **None**: no knowledge.
- **Familiar**: some knowledge, but not at application level.
- **Application**: used the technology
- **Expert**: extensive experience.

Programming languages		**None**	**Familiar**	**Application**	**Expert**
OO-languages:	Java				
	C++				
	Smalltalk				
Imperative languages:	C				
	Pascal				
	Basic				
	Modula				
	Fortran				
Functional languages:	Lisp				
	Miranda				
	Scheme				
	Haskell				
Logical languages:	Prolog				

Formal specification languages	**None**	**Familiar**	**Application**	**Expert**
First Order Logic				
Linear Temporal Logic				
Computational Tree Logic				
Future Interval Logic (FIL or GIL)				
Vienna Development Method				
Z				

Algebraic languages: OBJ2, Larch				
Other, specify:				

Specification diagrams	**None**	**Familiar**	**Application**	**Expert**
Finite state automata				
State charts				
Data flow diagrams				
Entity relationship diagrams				
Class diagrams, object models				
Collaboration and Sequence diagrams				
Petri nets				
Timeline diagrams				

Specification aspects	**None**	**Familiar**	**Application**	**Expert**
Invariants				
Assertions				
Requirements specifications				
Property classification				
Patterns and scopes				
Composite Propositions				
Design specification				

In what domains do you have software development experience?

Domain	**None**	**Familiar**	**Application**	**Expert**
Databases				
Transaction-oriented systems				
Real-time systems				
Communications				
Distributed systems				
Embedded systems				
Other				

Appendix C
Expirement Toturial

This appendix provides snapshots of the power-point presentation used in the tutorial of SPS and Prospec.

Property Specification Tutorial

Prospec Survey

Software Engineering Research Group
Systems and Software Engineering Affinity Lab
The University of Texas at El Paso

Outline

- Overview of Specification Pattern System (SPS)
  - Patterns
  - Scopes
  - Mappings to formal specification languages
- Examples
- Specifying properties
- Navigating Prospec

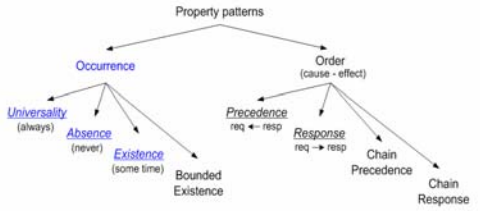
Specification Pattern System (SPS) Dwyer, et al. 1998

- Property: a description of the expected behavior of a program
  - A response shall be preceded by a request.
  - A train shall stop before it enters a closed gate.
- SPS
  - Provides a set of commonly occurring properties
  - Breaks down the specification into *pattern – scope*.
  - Assists practitioners in mapping properties to formal languages

Conditions and events

- Conditions
  - Assertions (*propositions*) are evaluated at a specific state.
  - Assertions can hold over more than one state.
  - Example: The temperature is within 60° F and 100° F.
  - Example: The train applies maximum braking.
- Events
  - Indicate the occurrence of a given circumstance.
  - Events do not have duration (defined or undefined).
  - Events can be associated with a condition.
    - The beginning or ending of a condition.
    - Example: When the temperature reaches 60°, the control shall be switched on.

Specification Pattern System – Dwyer et al., 1998



Occurrence Patterns

Universality: Property is true in every point of the execution.

Absence: property is never true during the execution.

Existence: property is true at some point in the execution.

Order Patterns

Precedence: *(P) precedes (T);* property requires that a given state or event (P) always occurs before a designated state or event (T) occurs.

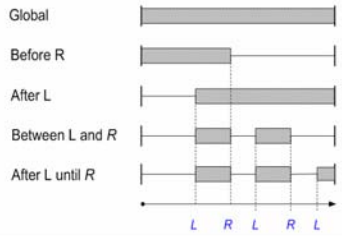
Response: *(T) responds to (P);* property requires that the occurrence of a given state or event (P) be followed by a designated state or event (S).

Chain precedence: (P) precedes (S,T): the occurrence of *S* followed by *T* must be preceded by an occurrence of *P* within the scope

Chain response: (S,T) responds to (P): the occurrence of the stimulus event must be followed by an occurrence of the sequence of response events within the scope.

Scope of the Property

- Defines the extent of the program execution over which the pattern must hold.
- Includes a starting and ending state/event (interval)



Scope Definitions

- **Global**: the entire program execution
- **Before**: the execution up to a given state/event
- **After**: the execution after a given state/event
- **Between**: any part of the execution from one given state/event to another given state/event
- **After-until**: similar to *between* except that the designated part of the execution continues even if the second state/event does not occur.

Steps for Specifying Properties

1. Identify propositions from the English description.
2. Assign symbols to each proposition.
3. Assign the symbols to pattern parameters P, T
   Universality (P), Absence (P), Existence (P)
   Precedence (P, T) and Response (P, T)
4. Assign the symbols to scope parameters L, R.

Example: ATM Withdraw Transaction

When a user approves a withdraw transaction, the ATM machine shall eventually terminate the following tasks: dispense the money, update the user's account, print the receipt, and return the ATM card.

U User has approved withdraw transaction.
M Money is dispensed.
A Account is updated.
P Receipt is printed.
C Card is returned.

Example: ATM Withdraw Transaction

Pattern: Response (S, P)

P : {U}
S : {M, A, P, C}

Scope: Global

Symbol assignment not needed.

Mapping:

$$\Box([\rightarrow U \mid \rightarrow) \Diamond(M \wedge A \wedge P \wedge C))$$

Multiple Propositions

The relations among multiple propositions define concurrent or sequential structures of behavior.

- Relationships
  - Concurrent
    - All the propositions hold at the same state.
    - One or more propositions hold.
  - Sequential
    - Propositions hold at different states.
    - Propositions hold at (non) consecutive states.
    - Propositions hold following a predefined order.

Example 2: BIOS

Once the BIOS has been loaded and the system has been initialized, the scheduler shall assign CPU time slice to user processes.

BL The BIOS is loaded.

SI System is initialized.

SR The scheduler is running.

a) *Scope:* After L *Pattern:* Universality ?

b) *Scope:* Global *Pattern:* Response ?

Example 2: BIOS

Pattern: *Universality (P)*

P : {SR}

Scope: *After L*

L : {BL, SI}

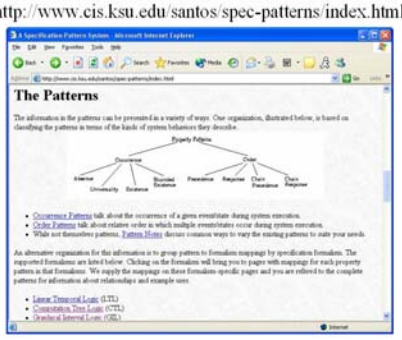
What is the relation among BL and SI?

Sequence? First BL and then SI

Concurrency? When BL and SI both hold

Navigating SPS

http://www.cis.ksu.edu/santos/spec-patterns/index.html



Patterns

Occurrence Specification Patterns

Occurrence patterns are used to express requirements related to the existence or lack of existence of certain states/events during well-defined regions of system execution. As with our other patterns, the regions are defined using Scopes.

Information about the patterns system as a whole is available at the Specification Patterns Home Page.

There are four occurrence patterns:

- Absence, aka Never
- Universality, aka Globally, Henceforth
- Existence, aka Eventually, Future
- Bounded Existence

http://www.cis.ksu.edu/santos/spec-patterns/occurrence.html

Order Specification Patterns

Order patterns are used to express requirements related to pairs of states/events during well-defined regions of system execution. As with our other patterns, the regions are defined using Scopes.

Information about the patterns system as a whole is available at the Specification Patterns Home Page.

There are two basic order-related patterns:

- Precedence
- Response, aka Follows, Leads-To

Chain patterns are used to express requirements related to complex combinations of individual state/event relationships. These include precedence/response relationships consisting of sequences of individual states/events. We call these chain patterns.

http://www.cis.ksu.edu/santos/spec-patterns/order.html

Absence Property Pattern

Intent

To describe a portion of a system's execution that is free of certain events or states. Also known as **Never**.

Examples and Known Uses

The most common example is mutual exclusion. In a state-based model, the scope would be global and *P* would be a state formula that is true if more than one process is in its critical section. For an event-based model, the scope would be a segment of the execution in which some process is in its critical section (i.e., between an enter section event and a leave section event), and *P* would be the event that some other process enters its critical section.

Relationships

This pattern is the dual of the Existence pattern. In fact, in many specification formalisms negation and explicit queries for existence will be used to formulate an instance of the Absence pattern, as seen in the examples above.

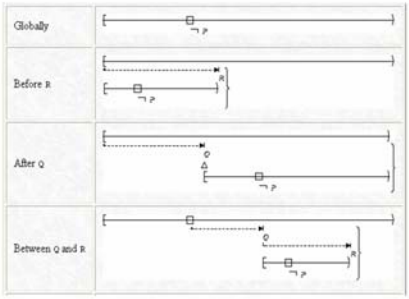
Note that Between scopes in this pattern (with a false proposition or empty event symbol) appear to be able to specify the same thing as a Response pattern with global scope. This is not the case, however, since the cause-effect relationship is required for the Response whereas the scope for this pattern is optional.

If one wishes to exclude states characterized by multiple propositions or multiple events one can do this by defining *P*

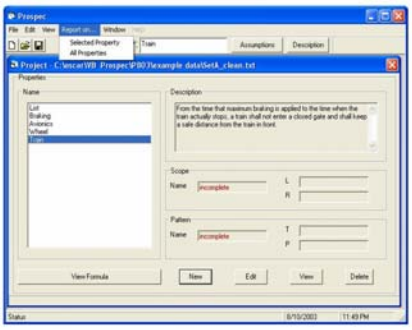
Mappings

- CTL
- LTL
- QRE
- BSCA
- SIL

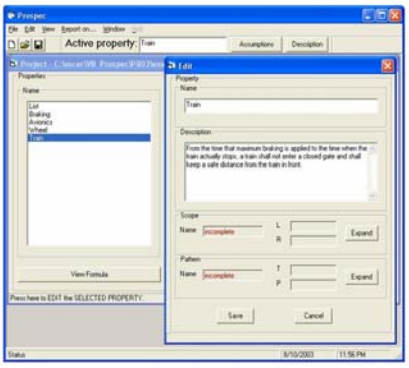
Mapping absence of P in GIL



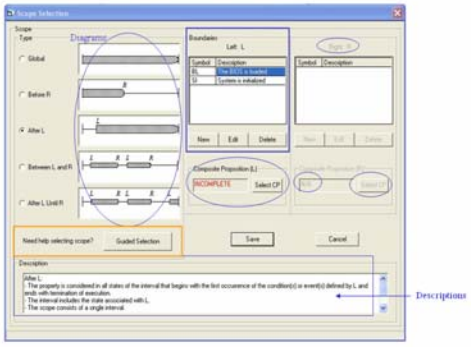
Property Specification and Summary



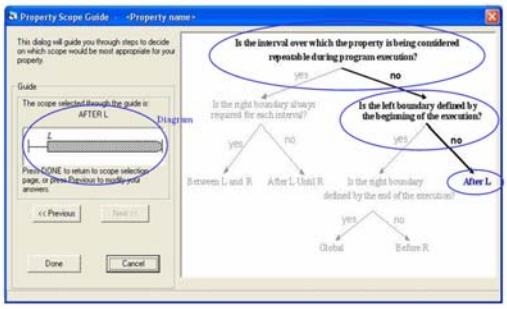
Edit Window



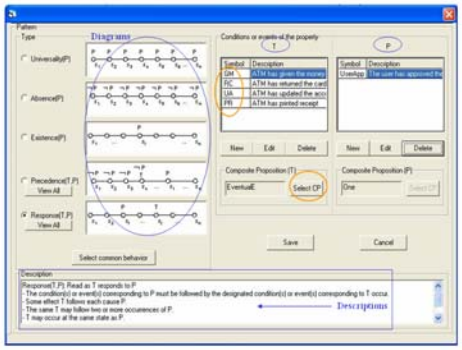
Scope Window



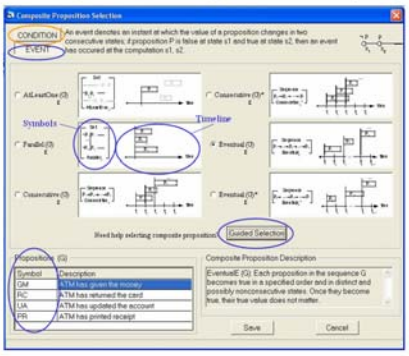
Scope Guidance



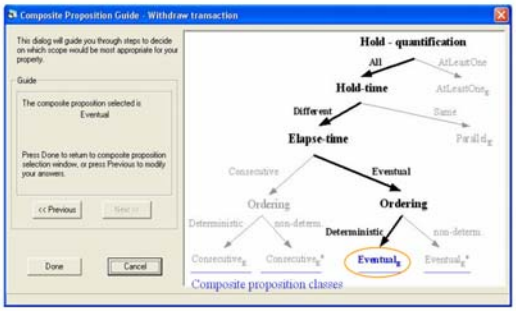
Pattern Window



Composite Propositions (CP)



Composite Proposition Guidance



Any questions?

Thank you.

Appendix D
In-Trial Forms

Form D-1: Prospec Property log*

Prospec Property Log

Subject ID:	Group ID:	Property ID: <set>-<#>

Property name: Emergency stop
Property Description: If a command expires, the train applies maximum braking until the train stops or a new command arrives.

Start time:_____
End time:_____

After using Prospec to specify the above property, please attach a copy of the specification by saving it to a text file and printing the file. This can be done by clicking on the "Report On" menu item and choosing the appropriate property. Before printing, make sure that the specification is complete. In addition, please answer the following questions:

1. In your opinion, is the English description of the property complete? ( )Yes ( ) No
   If your answer is "yes," skip question 2.

2. Write down any assumptions that you made about the property.

   _____
   _____
   _____
   _____

3. Mark the order in which you specified the property.
   ( ) Pattern first and scope second
   ( ) Scope first and pattern second

4. Mark the reason that best describes why you chose the specific order you marked in item 3 above.
   ( ) I do not have a reason for choosing one order over the other.
   ( ) I found it easier or more intuitive to specify the property in the order that I chose.
   ( ) Other_____
   _____

*This form was filled by the subjects for property A2. Similar forms were completed for each of the other properties.

Form D-2: SPS Property log*

SPS Property Log

Subject ID:	Group ID:	Property ID: <set>-<#>

Property name: Emergency stop
Property Description: If a command expires, the train applies maximum breaking until the train stops or a new command arrives.

Start time: _____ **End time**:_____

After using SPS to specify the above property, please answer the following questions.

1. In your opinion, is the English description of the property complete? ( )Yes ( ) No
   If your answer is "yes," skip question 2.

2. Write down any assumptions that you made about the property.
   _____
   _____
   _____
   _____

Questions 3 and 4 identify pattern and scope for the property being specified. If you are unable to identify pattern or scope, then enter "none" for the question. In the SPS Post-Evaluation form, explain what you would have liked to specify that you could not with the patterns or scopes provided.

3. Identify the propositions of the property and assign symbols to each of them.
   _____
   _____
   _____
   _____
   _____

4. Answer the following.
   a) Identify the desired pattern. _____
   b) If applicable, state the proposition(s) that apply to P. _____
   c) If applicable, state the proposition(s) that apply to S. _____
   d) If P and/or S have more than one proposition, state the relationship(s) among the propositions.
      P: _____
      S: _____
5. Answer the following.
   a) Identify the desired scope. _____
   b) If applicable, state the proposition(s) that apply to Q. _____
   c) If applicable, state the proposition(s) that apply to R. _____
   e) If Q and/or R have more than one proposition, state the relationship(s) among the propositions.
      Q: _____
      R: _____

*This form was filled by the subjects for property A2. Similar forms were completed for each of the other properties.

Appendix E
Assignment Logs

Form E-1 Group assignment log

Group Assignment Log

Date: __________

SUBJECT ID	GROUP ID

Form E-2: Subject assignment log

Subject Assignment Log
Institution: _____
Date: _____

ID NUMBER	NAME
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
14.	
15.	
16.	
17.	
18.	
19.	
20.	
21.	
22.	
23.	
24.	
25.	
26.	
27.	
28.	
29.	
30.	
31.	
32.	
33.	
34.	

Appendix F
Post Experiment Forms

Form F-1: Prospec Post experiment evaluation

Prospec Post-Evaluation Form

Subject ID:	Group ID:

Refer to the Instruction Supplement for assistance in terminology. For items 1-5, rank the level of help provided by Prospec as follows:

0 = none 1= not much help 2 = fair 3 = good 4 = excellent

1. Rank the level of help that the following provided for identifying the desired scope.
   Diagrams _____ Descriptions _____ Guidance (Wizard) _____

2. Rank the level of help that the following provided for identifying the desired pattern.
   Diagrams _____ Descriptions _____ Guidance (Wizard) _____

3. Rank the level of help that the following provided for identifying the relation among more than one propositions, i.e. *composite propositions* (CP).
   CP symbol_____ Timeline diagram _____ CP description _____ CP guidance _____

4. For *response pattern*, rank the level of help that the following provided for identifying the type of common concurrent behavior.
   CP symbol_____ Timeline_____ Petri-net_____ Description_____

5. If applicable, rank the level of help that the following provided for identifying incomplete descriptions of properties.
   Diagrams _____ Descriptions _____ Guidance (Wizard) _____

Answer the questions in each of the following boxes.

SCOPE

1. Enumerate any scope descriptions that need clarification.
   _____
   _____

2. Enumerate questions from the guidance/wizard that need clarification.
   _____
   _____
   _____

3. If the order of the questions in the guidance/wizard does not seem appropriate, describe the problem.
   _____
   _____

_____

4. If there is a scope(s) that you would have liked to specify but could not, describe the scope.

_____
_____
_____

PATTERN

1. Enumerate any pattern descriptions that need clarification.

_____
_____

2. If there are any diagrams that need clarification, please describe the problem.

_____
_____

3. If there is a pattern(s) that you would have liked to specify but could not, describe the pattern.

_____
_____
_____

COMPOSITE PROPOSITIONS

1. Enumerate any guidance/wizard descriptions that need clarification.

_____
_____
_____

2. If the order of the questions in the guidance/wizard does not seem appropriate, describe the problem.

_____
_____
_____

3. Enumerate any diagrams that need clarification.

_____
_____
_____

4. Describe any part of the descriptions for conditions and events that are unclear.

_____
_____

5. If there is a composite proposition(s) that you would have liked to specify but could not, describe the composite proposition.

_____
_____
_____

COMMENTS
Please remark on the navigation of Prospec for specifying a property and provide any other comments about the tool.

_____
_____
_____

Form F-2: SPS Post experiment evaluation

SPS Post-Evaluation

Subject ID: _____ Group ID: _____

1 Answer the questions in each of the following boxes.

SCOPES

1. Does the website provide the information that you need to understand the scopes? Please explain your answer.

_____
_____

2. Enumerate any of the descriptions that need clarification.

_____
_____

3. If there is a scope(s) that you would have liked to specify but could not, describe the scope.

_____
_____
_____

PATTERNS

1. Does the website provide the information that you need to understand the patterns? Please explain your answer.

_____
_____

2. Enumerate any of the descriptions that need clarification.

_____
_____

3. If there is a pattern(s) that you would have liked to specify but could not, describe the pattern.

_____
_____
_____

COMMENTS
Please remark on the navigation of SPS for specifying a property and provide any other comments about SPS.

_____
_____

Appendix G
Verification of the Semantics for SPS Patterns and Scopes.

The semantics was verified with the SPIN model checker. There is a test suite for the following patterns: universality, absence, existence, response, and precedence. Each test suite includes several tests for each of the scopes associated to a given pattern. Table G1 describes the elements considered in the test suite documentation.

Table G1: Description of the test suit elements.

ELEMENT	DESCRIPTION
TEST #	The test number in the test suite. Additionally, it describes the trace of computation to be tested. The trace is read as follows. A dash (-) implies that none of the propositions is *true* at that state. A letter symbol implies that the proposition is valid at that state. Displaying two letters (one on top of the other) implies that both propositions are valid at that state. For instance, in trace : - - *p* - - *r* – *t* no proposition is valid on states 0, 1, 3, 4, and 6. Let *p, t,* and *r* denote propositions, then *p* holds in state 2 and *r* and *t* hold in state 5.
PATTERN	Names the pattern being tested, e.g., *Absence* of (*p*), *Existence* of (*p*), *Universality* of (*p*), *Precedence* (*t* precedes *p*), *Response* (*t* responds to *p*).
SCOPE	Names the pattern being tested, e.g., Global, Before *R*, After *L*, Between *L* and *R*, After *L* until *R*.
FORMULA	The LTL formula describing the pattern and scope. This formula was tested by the SPIN model checker. Because SPIN does not support the *weak until* (***W***) operator, the LTL equivalence was used instead.
P, T, L, R	Provides a description of the propositions as they were entered in the SPIN LTL property panel. For example *P*: (*i* = = 0) asserts that *P* is *true* when program variable *i* is equal to 0. Depending on the pattern and scope, some of these propositions might not be used in the test case. For example, in a *response* pattern with a *global* scope, only *P* and *T* are present.
INTERVAL	Specifies whether the interval defined by the scope can be built in the trace of computation given by the test case. Consider the scope Between *L* and *R*. For trace - - *L* - *P* - *R* -, the interval is built and for trace - *L* - *P* - -, the interval is not built.

EXPECTED RESULT	Result expected by visual inspection.
ACTUAL RESULT	Result returned by the SPIN model checker.

Table G2: Test suite for the *absence* pattern.

Test #1 : - - - - - - - - - - **Pattern** : Absence (p) **Scope** : Global **Formula**: ¬ (◊ P) **P:** (i > limit + 1) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #2** : P P P P P P P P P P **Pattern** : Absence (p) **Scope** : Global **Formula**: ¬ (◊ P) **P**: (i <= limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #3 : - - - - - - - - - - P **Pattern** : Absence (p) **Scope** : Global **Formula**: ¬ (◊ P) **P** : (i == limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #4** : P - - - - - - - - - **Pattern** : Absence (p) **Scope** : Global **Formula**: ¬ (◊ P) **P** : (i = 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #5 : - - - - - - - - - - R **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i < 0) **R** : (i == limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #6** : P - - - - - - - - - R **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i == 0) **R** : (i == limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #7 : - - - - - - - - - - **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i < 0) **R** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #8** : - - - R P P P P P P **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i >= 3) **R** : (i == 3) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #9 : R - - - - - - - - - **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i < 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #10** : R P P P P P P P P P P **Pattern** : Absence (p) **Scope** : Before R **Formula**: ◊R → (¬ (¬R *U* P)) **P** : (i >= 0) **R** : (i == 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #11 : L - - - - - - - - - **Pattern** : Absence (p)	**Test #12** : P - - - - - - - - L

Scope : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i < 0) **L** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i <= 0) **L**: (i = = 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #13 : P P P - - - - - - - L **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i <= 2) **L** : (i = = 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #14** : P P L - - - - - - - **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i < = 1)) **L** : (i = = 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #15 : - L P - - - - - - - - - **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i = = 2) **L** : (i = = 1) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #16** : - - L - P - - - - - **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i = = 4) **L** : (i = = 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #17 : P P P P P P P P P P **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i >= 0) **L** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: Violation	**Test #18** : - - - - - - - - - - - **Pattern** : Absence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ¬(◊ P)) **Formula**: □¬L || (¬L *U* (L ∧ ¬(◊ P)) **P** : (i < 0) **L** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #19 : - - - - - - - - - - - **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i < 0) **L** : (i > limit + 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #20** : R P P P P - - - L **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i > 0) && (i < 6) **L** : (i = = limit) **R**: (i = = 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #21 : - L - - P - - - - - **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i = = 4)	**Test #22** : L - - - - - - - - - P R **Pattern** : Absence (p) **Scope** : Between L - R

L : (i = = 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i == 0) **L** : (i = = 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #23 : L - P - R - - - - - **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i == 2) **L** : (i = = 0) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #24** : L - - - - - - - - R P **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i == 0) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #25 : L- - - - - - - - - R **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i < 0) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #26** : - - - L R - - - - - P **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i = = 4) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #27 : - - - L R - - - - - P **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i = = 3) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #28** : - - L - - R - L - P - **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i == limit - 1) **L** : (i = = 2) || (i = = 7) **R** : (i = = 5) **Interval**: Yes … No **Expected Result**: No violation **Actual Result**: No violation
Test #29 : - L - - R - L P - R **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : (i = 7) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #30** : - L - - R - L - - R **Pattern** : Absence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬ (¬R *U* P)) **P** : ((i > i) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #31 : - - - - - - - - - - **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i >limit + 1)	**Test #32** : - - P - - - - R - - **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i ==2)

L : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**L** : (i == limit + 2) **R** : (i = =7) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #33 : P P P P P P P P P P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i >= 0) **L** : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #34** : - L - - - - - - - - R P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i == 1) **L** : (i == 1) **R** : (i = =limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #35 : L - - - - - - - - R **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P**: (i < 0) **L** : (i == 0) **R** : (i = =limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #36** : - L - - - - - - - P R **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i < limit - 1) **L** : (i == 1) **R** : (i = =limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #37 : - L R - - - - - - - - P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P**: (i == 1) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #38** : - L R - - - - - - - - P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i ==2) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #39 : L - - - - - - - - - P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P**: (i == limit) **L** : (i == 0) **R** : (i < 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #40** : L - - - - - - - - - - **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i < 0) **L** : (i == 0) **R** : (i > i) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #41 : L - - R - L - P P - **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i == 7) || (i == 8) **L** : ((i == 0)|| (i == 5))	**Test #42** : L - - R - L - - - - **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i < 0) **L** : ((i == 0)|| (i == 5))

R : (i == 3)) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**R** : (i == 3)) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #43 : L - - R - L - - - R **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i < 0) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #44** : L - - R - - L - - - R P **Pattern** : Absence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (¬R *U* P)) **P** : (i == 5) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation

Table G3. Test suite for the *existence* pattern.

Test #1 : - - - - - - - - - - **Pattern** : Existence (p) **Scope** : Global **Formula**: ◊P→(**P:** (i > limit + 1) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #2** : - - - - - P - - - - **Pattern** : Existence (p) **Scope** : Global **Formula**: ◊P **P**: (i == 5) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #3 : - - - - - - - - - P **Pattern** : Existence (p) **Scope** : Global **Formula**: ◊P **P** : (i == limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #4** : P - - - - - - - - - **Pattern** : Existence (p) **Scope** : Global **Formula**: ◊P **P** : (i == 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #5 : - - - - P - - - R **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i == 4) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #6** : - - - - R P - - - **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i == 5) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #7 : - - - P - - - - - R **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i == 3) **R** : (i == 3) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #8** : P P P P P P P P P **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i >= 0) **R** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation

Test #9 : R - - - - - - - - - - **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i < 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #10** : R P P P P P P P P P **Pattern** : Existence (p) **Scope** : Before R **Formula**: ◊R → (¬(R) (P *U* ¬R))) **P** : (i > 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #11 : L P P P P P P P P P **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i > 0) **L** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #12** : - - - - - - - - - L **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i < 0) **L**: (i == limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #13 : P P P P L - - - - **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i <= 3) **L** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #14** : - - -P - - - - - - L **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i == 3) **L** : (i == 3) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #15 : - L P - - - - - - - - - **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i = = 2) **L** : (i = = 1) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #16** : - - L - P - - - - - **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i = = 4) **L** : (i = = 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #17 : P P P P P P P P P P **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i >= 0) **L** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No Violation	**Test #18** : - - - - - - - - - - **Pattern** : Existence (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ ◊P) **Formula**: □¬L || (¬L *U* (L ∧ ◊P)) **P** : (i < 0) **L** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #19 : - - - - - - - - - - **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i < 0)	**Test #20** : R - - - - - - - - L **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i < 0)

L : (i > limit + 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**L** : (i = = limit) **R**: (i = = 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #21 : - L - - - - - - - - **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i < i) **L** : (i = = 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #22** : L - - - - - - - - - R **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i < 0) **L** : (i = = 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #23 : L - - - R - - - - - **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i < 0) **L** : (i = = 0) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #24** : L - - - -P - - - - R **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i == 4) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #25 : P - - - - - - - - R L **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i == 0) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #26** : - - - L R - - - - - P **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i = = 4) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #27 : - - - L R - - - - - P **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i = = 3) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #28** : - - L - P R - L - - **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i ==4) **L** : (i = = 2) || (i = = 7) **R** : (i = = 5) **Interval**: Yes … No **Expected Result**: No violation **Actual Result**: No violation
Test #29 : - L P P R - L - - R **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (((i >= 1) && (i < 4)) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4)	**Test #30** : - L P - R - L P - R **Pattern** : Existence (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (¬(R) *U* (P ∧ ¬(R)))) **P** : (i == 2 || i == 7)) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4)

Interval: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #31 : - - - - - - - - - - **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i >limit + 1) **L** : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #32** : - - - - - - - R - - **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i > i) **L** : (i == limit + 2) **R** : (i = =7) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #33 : P P P P P P P P P P **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i >= 0) **L** : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #34** : - L P P P P P P P R **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i > 1) && (i < limit) **L** : (i == 1) **R** : (i = =limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #35 : P- - - - - - - - - R L **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P**: (i == 0) **L** : (i == 0) **R** : (i = =limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #36** : P P L - - - - - R **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i <= 1) **L** : (i == 2) **R** : (i = =limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #37 : - L R - - - - - - - - P **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P**: (i == 1) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #38** : - L R - - - - - - - - P **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i ==2) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #39 : P - - - - - - - - - L **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P**: (i == 0) **L** : (i == 0) **R** : (i < 0) **Interval**: Yes **Expected Result**: No violation	**Test #40** : L - P - - - - - - - - **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i == 2) **L** : (i == 0) **R** : (i < 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation

Actual Result: No violation	
Test #41 : L P - R - L - - - - **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i == 1) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #42** : L P - R - L - - -P **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : ((i == 1) || (i == 5) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #43 : L - P R - L - - P R **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : ((i == 2) || (i == limit – 1) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #44** : L P - R - L - - - R **Pattern** : Existence (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (¬ (R) *U* (P ∧ ¬ (R)))) **P** : (i == 1) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation

Table G4: Test suite for the *universality* pattern.

Test #1 : - - - - - - - - - - **Pattern** : Universality (p) **Scope** : Global **Formula**: □ P **P:** (i > limit + 1) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #2** : P P P P P P P P P P **Pattern** : Universality (p) **Scope** : Global **Formula**: □ P **P**: (i <= limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #3 : - P P P - - - - - P **Pattern** : Universality (p) **Scope** : Global **Formula**: □ P **P** : (((i > 0) && (i < 4)) || (i == limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #4** : - P P P P P P P P - **Pattern** : Universality (p) **Scope** : Global **Formula**: □ P **P** : ((i != 0) && (i != limit)) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #5 : P P P P P P P P P R **Pattern** : Universality (p) **Scope** : Before R **Formula**: ◊R → (P *U* R) **P** : (i < Limit) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #6** : - P P P P P P P P R **Pattern** : Universality (p) **Scope** : Before R **Formula**: ◊R → (P *U* R) **P** : (i < Limit) && (i > 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #7 : - - - - - - - - - - **Pattern** : Universality (p) **Scope** : Before R	**Test #8** : P P P P P P P P P **Pattern** : Universality (p) **Scope** : Before R

Formula: ◊R → (P *U* R) **P** : (i < 0); **R** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Formula**: ◊R → (P *U* R) **P** : (i >= 0) ; **R** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #9 : R - - - - - - - - - **Pattern** : Universality (p) **Scope** : Before R **Formula**: ◊R → (P *U* R) **P** : (i < 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #10** : R P P P P P P P P P **Pattern** : Universality (p) **Scope** : Before R **Formula**: ◊R → (P *U* R) **P** : (i > 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #11 : L P P P P P P P P **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i > 0) **L** : (i = = 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #12** : P P P P P P P P P P L **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i >= 0) **L**: (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #13 : P P P P P P P P P P L **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i >= 0) **L** : (i = = 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #14** : P P P P P P - - P L **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : ((i < 7) || (i == limit)) **L** : (i = = 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #15 : - - P P P P P P P P L **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i >= 2) **L** : (i = = 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #16** : - - L P P P P P P P **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i > 2) **L** : (i = = 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #17 : - L P - - - - - - - - - **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i = = 2) **L** : (i = = 1)	**Test #18** : - - L - P - - - - - **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i = = 4) **L** : (i = = 2)

Interval: Yes **Expected Result**: Violation **Actual Result**: Violation	**Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #19 : P P P P P P P P P P **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i >= 0) **L** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: Violation	**Test #20** : - - - - - - - - - - **Pattern** : Universality (p) **Scope** : After L **Formula**: ¬(L) *W* (L ∧ □P) **Formula**: □¬L || (¬L *U* (L ∧ □P)) **P** : (i < 0) **L** : (i > limit + 1) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #21 : - - - - - - - - - - **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i < 0) **L** : (i > limit + 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #22** : R P P P P - - - L **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i > 0) && (i < 6) **L** : (i = = limit) **R**: (i = = 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #23 : - L - - P - - - - - **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i = = 4) **L** : (i = = 1) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #24** : L - - - - - - - - - R **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i < 0) **L** : (i = = 0) **R** : (i = = 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #25 : L - - - R - - - - - **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i < 0) **L** : (i = = 0) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #26** : L P P P P P P P P R **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i > 0) && (i < limit) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #27 : P P P P P P P P P R L **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i >= 0) && (i < limit) **L** : (i = = 0) **R** : (i = = limit) **Interval**: Yes **Expected Result**: No violation	**Test #28** : - - - L R - - - - - P **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i = = 4) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: Violation

Actual Result: No violation	**Actual Result**: Violation
Test #29 : - - - L R - - - - - P **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i = = 3) **L** : (i = = 3) **R** : (i = = 4) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #30** : - - L P P R - L - - P **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (i > 1) && (i < 4) **L** : (i = = 2) || (i = = 7) **R** : (i = = 4) **Interval**: Yes … No **Expected Result**: No violation **Actual Result**: No violation
Test #31 : - L P P R - L P - R P **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : (((i >= 1) && (i < 4)) || (i = 7)) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #32** : - L P P R - L P P R P P **Pattern** : Universality (p) **Scope** : Between L - R **Formula**: □ ((L ∧ ¬(R) ∧ ◊R) → (P *U* R) **P** : ((i >= 1) && (i < 4)) || ((i = 7) && (i < limit)) **L** : ((i == 1) || (i == 6)) **R** : (i == limit) || (i = = 4) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #33 : - - - - - - - - - - **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i >limit + 1) **L** : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #34** : - - P - - - - R - - **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i ==2) **L** : (i == limit + 2) **R** : (i = =7) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation
Test #35 : P P P P P P P P P P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i >= 0) **L** : (i == limit + 2) **R** : (i < 0) **Interval**: No **Expected Result**: No violation **Actual Result**: No violation	**Test #36** : - L P P P P P P P R **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i > 1) && (i < limit) **L** : (i == 1) **R** : (i = =limit) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #37 : P P P P P P P P P R L **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P**: (i >= 0) && (i < limit) **L** : (i == 0) **R** : (i = =limit)	**Test #38** : - L P P P P P P - R **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i > 1) && (i < limit - 1) **L** : (i == 1) **R** : (i = =limit) **Interval**: Yes

Interval: Yes **Expected Result**: No violation **Actual Result**: No violation	**Expected Result**: Violation **Actual Result**: Violation
Test #39 : - L R - - - - - - - - P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P**: (i == 1) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #40** : - L R - - - - - - - - P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i ==2) **L** : (i == 1) **R** : (i == 2) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation
Test #41 : P P P P P P P P - - L **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P**: (i >= 0) && (i < limit – 1) **L** : (i == 0) **R** : (i < 0) **Interval**: Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #42** : P P P P P P P P P P L **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (i >= 0) **L** : (i == 0) **R** : (i < 0) **Interval**: Yes **Expected Result**: No violation **Actual Result**: No violation
Test #43 : L P P R – L P - - P P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : (((i >= 0)&&(i < 3)) || (i == 6) || (i == limit)) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation	**Test #44** : L P P R – L P P P P P P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : ((i >= 0)&&(i < 3)) || (i >= 5) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation
Test #47 : L P P R – L P P P R P P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : ((i >= 0)&&(i < 3)) || ((i >= 5) && (i < limit) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: No violation **Actual Result**: No violation	**Test #48** : L P P R - - L P P P R P **Pattern** : Universality (p) **Scope** : After L – Until R **Formula**: □ (L ∧ ¬ (R) → (P *W* R)) **Formula**: □((L ∧ ¬(R)) → (□P || (P U R))) **P** : ((i >= 0)&&(i < 3)) || ((i > 5) && (i < limit) **L** : ((i == 0)|| (i == 5)) **R** : (i == 3)) || (i == limit) **Interval**: Yes … Yes **Expected Result**: Violation **Actual Result**: Violation

Table G5: Test suite for the *precedence* pattern.

Test #1 : - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i > limit + 1) **T:** (i < 0) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #2** : - - - T - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i > limit + 1) **T:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #3 : P- - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i == 0) **T:** (i > limit + 1) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not Valid	**Test #4** : - - -T - - - - - - P **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i == 3) **T:** (i == 3) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid
Test #5 : - - - P - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i == 3) **T:** (i > limit + 1) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #6** : - P - T - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Global **Formula:** ¬ (P) *W* (T ∧ ¬ (P)) **Formula:** (□¬(P)) ∨ ((¬P) *U* (T ∧ ¬P)) **P:** (i == 1) **T:** (i == 3) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid
Test #7 : - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** ◊R → (¬P *U* ((T ∧ ¬P) ∨ R)) **P:** (i > limit + 1) **T:** (i < 0); **R:** (i < 0) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #8** : - - - - - P - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** ◊R → (¬P *U* ((T ∧ ¬P) ∨ R)) **P:** (i == 5) **T:** (i < 0); **R:** (i > limit + 1) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #9 : - - - - - - R - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** ◊R → (¬P *U* ((T ∧ ¬P) ∨ R)) **P:** (i > limit + 1) **T:** (i < 0); **R:** (i == 6) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #10** : - - - - - - R - P - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** ◊R → (¬P *U* ((T ∧ ¬P) ∨ R)) **P:** (i == 8) **T:** (i < 0); **R:** (i == 6) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #11 : - - - P - - R - - **Pattern** : Precedence : (T) Precedes (P)	**Test #12** : R - - - - - - - - **Pattern** : Precedence : (T) Precedes (P)

Scope : Before R **Formula:** $\Diamond R \rightarrow (\neg P \; U \; ((T \wedge \neg P) \vee R))$ **P:** (i == 3) **T:** (i < 0) **R:** (i == 6) **Interval**: Built **Expected Result**: Not Valid **Actual Result**: Not valid	**Scope** : Before R **Formula:** $\Diamond R \rightarrow (\neg P \; U \; ((T \wedge \neg P) \vee R))$ **P:** (i > limit + 1) **T:** (i < 0) **R:** (i == 0) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #13 : - T- - P - - R - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** $\Diamond R \rightarrow (\neg P \; U \; ((T \wedge \neg P) \vee R))$ **P:** (i == 4) **T:** (i == 1) **R:** (i == 7) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #14** : - - - P - - R - - T **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** $\Diamond R \rightarrow (\neg P \; U \; ((T \wedge \neg P) \vee R))$ **P:** (i == 3) **T:** (i == 3) **R:** (i == 7) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid
Test #15 : - - - P - - - - - R T **Pattern** : Precedence : (T) Precedes (P) **Scope** : Before R **Formula:** $\Diamond R \rightarrow (\neg P \; U \; ((T \wedge \neg P) \vee R))$ **P:** (i == 3) **T:** (i == 3) **R:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #16** : - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg (L) \; W \; (L \wedge (\neg P \; W \; (T \wedge \neg P)))$ **Formula:** $\Box(\neg L) \vee \neg L \; U \; (L \wedge (\Box\neg P \vee \neg P U \; (T \wedge \neg P)))$ **P:** (i < 0) **T:** (i > limit + 1) **L:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #17 : - - - L - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg (L) \; W \; (L \wedge (\neg P \; W \; (T \wedge \neg P)))$ **Formula:** $\Box(\neg L) \vee \neg L \; U \; (L \wedge (\Box\neg P \vee \neg P U \; (T \wedge \neg P)))$ **P:** (i < 0) **T:** (i > limit + 1) **L:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #18** : -P - L - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg (L) \; W \; (L \wedge (\neg P \; W \; (T \wedge \neg P)))$ **Formula:** $\Box(\neg L) \vee \neg L \; U \; (L \wedge (\Box\neg P \vee \neg P U \; (T \wedge \neg P)))$ **P:** (i == 2) **T:** (i > limit + 1) **L:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #19 : - - - L - - - - - P T **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg (L) \; W \; (L \wedge (\neg P \; W \; (T \wedge \neg P)))$ **Formula:** $\Box(\neg L) \vee \neg L \; U \; (L \wedge (\Box\neg P \vee \neg P U \; (T \wedge \neg P)))$ **P:** (i == 3) **T:** (i == 3) **L:** (i == 3) **Interval**: Built **Expected Result**: Not valid	**Test #20** : - - L - - P - T- - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg (L) \; W \; (L \wedge (\neg P \; W \; (T \wedge \neg P)))$ **Formula:** $\Box(\neg L) \vee \neg L \; U \; (L \wedge (\Box\neg P \vee \neg P U \; (T \wedge \neg P)))$ **P:** (i == 5) **T:** (i == 7) **L:** (i == 2) **Interval**: Built **Expected Result**: Not Valid **Actual Result**: Not valid

Actual Result: Not valid	
Test #21 : - L -- T- - P- - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg$ (L) *W* (L $\wedge$ ($\neg$P *W* (T $\wedge$ $\neg$P))) **Formula:** $\Box(\neg L) \vee \neg L$ *U* (L $\wedge$ ($\Box\neg P \vee \neg P U$ (T $\wedge$ $\neg$P))) **P:** (i == 7) **T:** (i == 4) **L:** (i == 1) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #22** : - - L - - P - - - - T **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L **Formula:** $\neg$ (L) *W* (L $\wedge$ ($\neg$P *W* (T $\wedge$ $\neg$P))) **Formula:** $\Box(\neg L) \vee \neg L$ *U* (L $\wedge$ ($\Box\neg P \vee \neg P U$ (T $\wedge$ $\neg$P))) **P:** (i == 5); **T:** (i == 5) **L:** (i == 2) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid
Test #23 : - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i > limit + 1) **T:** (i < 0) **L:** (i < i) **R:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #24**: - - - P - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 3) **T:** (i < 0) **L:** (i < i) **R:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #25 : - - - R- - - P- - L - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 7) **T:** (i < 0) **L:** (i == 10) **R:** (i == 3) **Interval**: Not Built **Expected Result**: Valid **Actual Result**: Valid	**Test #26** : - L- -P - T- - R - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 4) **T:** (i == 6) **L:** (i == 1) **R:** (i == 9) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid
Test #27 : - - - R - L - P - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 7) **T:** (i < 0) **L:** (i == 5) **R:** (i == 3) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #28** : - - R - P - -L - - T **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 4); **T:** (i == 4) **L:** (i == 7); **R:** (i == 2) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #29 : - - L - P - -R - - T **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 4) **T:** (i == 4)	**Test #30** : - L T - P R - L - P **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** $\Box$ (($L \wedge \neg R \wedge \Diamond R$) $\rightarrow$ ($\neg$P *U* ((T $\wedge$ $\neg$P) $\vee$ R) **P:** (i == 4 || i == 9) **T:** (i == 2)

L: (i == 2) **R:** (i == 7) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**L:** (i == 1 || i == 7) **R:** (i == 5) **Interval**: Built ... Not built **Expected Result**: Valid **Actual Result**: Valid
Test #31 : - L T - P R - L – P -R **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** □ ((L ∧ ¬R ∧ ◊R) → (¬P *U* ((T ∧ ¬P) ∨ R) **P:** (i == 4 || i == 9) **T:** (i == 2) **L:** (i == 1 || i == 7) **R:** (i == 5 || i == 11) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #32** : - L T - P R - L T P -R **Pattern** : Precedence : (T) Precedes (P) **Scope** : Between L and R **Formula:** □ ((L ∧ ¬R ∧ ◊R) ¬ (¬P *U* ((T ∧ ¬P) ∨ R) **P:** (i == 4 || i == 9) **T:** (i == 2 || i == 8) **L:** (i == 1 || i == 7) **R:** (i == 5 || i == 11) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid
Test #33 : - - - - - - - - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i < 0) **T:** (i < i)) **L:** (i > i) **R:** (i > limit + 1) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #34** : - - -P - - L - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 3) **T:** (i < i)) **L:** (i == 6) **R:** (i > limit + 1) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #35 : - -L - P T - - R - - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 4); **T:** (i == 5)) **L:** (i == 2); **R:** (i == 8) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #36** : - -L - T - P - R - - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 6); **T:** (i == 4)) **L:** (i == 2); **R:** (i == 8) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #37 : - -L - T - R - - - - P **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 6); **T:** (i == 6) **L:** (i == 2); **R:** (i == 8) **Interval**: Built **Expected Result**: Not valid **Actual Result**:	**Test #38** : - -L - R - - P- - - - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 7); **T:** (i > i) **L:** (i == 2); **R:** (i == 4) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid

Test #39 : - L T P R - - L - - P - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 3 || i == 10) **T:** (i == 2) **L:** (i == 1 || i == 7) **R:** (i == 4) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #40** : - L T P R - - L T - P - **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 3 || i == 10) **T:** (i == 2 || i == 8) **L:** (i == 1 || i == 7) **R:** (i == 4) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid
Test #41 : - L T P R - - L T - P R **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 3 || i == 10) **T:** (i == 2 || i == 8) **L:** (i == 1 || i == 7) **R:** (i == 4 || i == 11) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid	**Test #42** : - L T P R - - L - - P R **Pattern** : Precedence : (T) Precedes (P) **Scope** : After L Until R **Formula:** □ (L ∧ ¬R→(¬P *W* ((T ∧ ¬P) ∨ R))) **Formula:** □ (L ∧ ¬R→(□¬P∨ (¬P *U* ((T ∧ ¬P) ∨ R)))) **P:** (i == 3 || i == 10) **T:** (i == 2) **L:** (i == 1 || i == 7) **R:** (i == 4 || i == 11) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid

Table G6: Test suite for the *response* pattern.

Test #1 : - - - - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i > limit + 1) **T:** (i < 0) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #2** : - - - T - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i > limit + 1) **T:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #3 : - - - - - - - - - P **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i == limit) **T:** (i > limit + 1) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #4** : - - -T - - - - - - P **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i == 3) **T:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #5 : - - - P - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i == 3) **T:** (i > limit + 1) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #6** : - P - T - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Global **Formula:** □(P→ ◊T) **P:** (i == 1) **T:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #7 : - - - - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i > limit + 1) **T:** (i < 0) **R:** (i < 0) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #8** : - - - - - P - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 5) **T:** (i < 0) **R:** (i > limit + 1) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #9 : - - - - - - R - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i > limit + 1) **T:** (i < 0) **R:** (i == 6) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #10** : - - - - - - R - P - **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 8) **T:** (i < 0) **R:** (i == 6) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #11 : - - - P - - R - - **Pattern** : Response: (T) Responds to (P)	**Test #12** : R - - - - - - - - **Pattern** : Response: (T) Responds to (P)

Scope : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 3) **T:** (i < 0) **R:** (i == 6) **Interval**: Built **Expected Result**: Not Valid **Actual Result**: Not valid	**Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i > limit + 1) **T:** (i < 0) **R:** (i == 0) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #13 : - P- - T - - R - - **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 1) **T:** (i == 4) **R:** (i == 7) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #14** : - - - P - - R - - T **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 3) **T:** (i == 3) **R:** (i == 7) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #15 : - - - P - - - - - R **Pattern** : Response: (T) Responds to (P) **Scope** : Before R **Formula:** ◊ R → ((P →((¬R) *U* (T ∧ ¬R)))) *U* R) **P:** (i == 3) **T:** (i > i) **R:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #16** : - - - - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** ¬ (L) *W* (L ∧ □ (P → ◊ T)) **Formula:** □(¬L) ∨ (¬L *U* (L ∧ □ (P → ◊ T))) **P:** (i < 0) **T:** (i > limit + 1) **L:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #17 : - - - L - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** ¬ (L) *W* (L ∧ □ (P → ◊ T)) **Formula:** □(¬L) ∨ (¬L *U* (L ∧ □ (P → ◊ T))) **P:** (i < 0) **T:** (i > limit + 1) **L:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #18** : -T - L - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** ¬ (L) *W* (L ∧ □ (P → ◊ T)) **Formula:** □(¬L) ∨ (¬L *U* (L ∧ □ (P → ◊ T))) **P:** (i < 0) **T:** (i == 1) **L:** (i == 3) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #19 : - - - L - - - - - P T **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** ¬ (L) *W* (L ∧ □ (P → ◊ T)) **Formula:** □(¬L) ∨ (¬L *U* (L ∧ □ (P → ◊ T))) **P:** (i == 3) **T:** (i == 3) **L:** (i == 3) **Interval**: Built **Expected Result**: Valid	**Test #20** : - - L - - T - P- - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** ¬ (L) *W* (L ∧ □ (P → ◊ T)) **Formula:** □(¬L) ∨ (¬L *U* (L ∧ □ (P → ◊ T))) **P:** (i == 7) **T:** (i == 5) **L:** (i == 2) **Interval**: Built **Expected Result**: Not Valid **Actual Result**: Not valid

Actual Result: Valid	
Test #21 : - L - - P- - T- - **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** $\neg$ (L) *W* (L $\wedge$ $\Box$ (P $\rightarrow$ $\Diamond$ T)) **Formula:** $\Box(\neg L) \vee (\neg L \; U \; (L \wedge \Box (P \rightarrow \Diamond T)))$ **P:** (i == 4) **T:** (i == 7) **L:** (i == 1) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**Test #22** : - - L - - P - - - - T **Pattern** : Response: (T) Responds to (P) **Scope** : After L **Formula:** $\neg$ (L) *W* (L $\wedge$ $\Box$ (P $\rightarrow$ $\Diamond$ T)) **Formula:** $\Box(\neg L) \vee (\neg L \; U \; (L \wedge \Box (P \rightarrow \Diamond T)))$ **P:** (i == 5) **T:** (i == 5) **L:** (i == 2) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #23 : - - - - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i > limit + 1) **T:** (i < 0) **L:** (i < i) **R:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #24**: - - - P - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 3) **T:** (i < 0) **L:** (i < i) **R:** (i > i) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #25 : - - - R- - - P- - L - **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 7) **T:** (i < 0) **L:** (i == 10) **R:** (i == 3) **Interval**: Not Built **Expected Result**: Valid **Actual Result**: Valid	**Test #26** : - L- -P - T- - R - **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 4) **T:** (i == 6) **L:** (i == 1) **R:** (i == 9) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #27 : - - - R - L - P - - **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 7) **T:** (i < 0) **L:** (i == 5) **R:** (i == 3) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #28** : - - R - P - -L - - T **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 4) **T:** (i == 4) **L:** (i == 7) **R:** (i == 2) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid
Test #29 : - - L - P - -R - - T **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 4); **T:** (i == 4)	**Test #30** : - L P - T R - L - P **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** $\Box$ ((L $\wedge \neg$R $\wedge \Diamond$R)$\rightarrow$ (P $\rightarrow$ ($\neg$R *U* (T$\wedge \neg$R))))*U* R) **P:** (i == 2 || i == 9) **T:** (i == 4) **L:** (i == 1 || i == 7)

L: (i == 2); **R:** (i == 7) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid	**R:** (i == 5) **Interval**: Built ... Not built **Expected Result**: Valid **Actual Result**: Valid
Test #31 : - L P - T R - L – P -R **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** □ ((L ∧ ¬R ∧ ◊R)→ (P → (¬R *U* (T∧ ¬R))))*U* R) **P:** (i == 2 || i == 9) **T:** (i == 4) **L:** (i == 1 || i == 7) **R:** (i == 5 || i == 11) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #32** : - L P - T R - L P T -R **Pattern** : Response: (T) Responds to (P) **Scope** : Between L and R **Formula:** □ ((L ∧ ¬R ∧ ◊R)→ (P → (¬R *U* (T∧ ¬R))))*U* R) **P:** (i == 2 || i == 8) **T:** (i == 4 || i == 9) **L:** (i == 1 || i == 7) **R:** (i == 5 || i == 11) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid
Test #33 : - - - - - - - - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i < 0) **T:** (i < i)) **L:** (i > i) **R:** (i > limit + 1) **Interval**: Not built **Expected Result**: Valid **Actual Result**: Valid	**Test #34** : - - -P - - L - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 3) **T:** (i < i)) **L:** (i == 6) **R:** (i > limit + 1) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #35 : - -L - T P - - R - - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 5) **T:** (i == 4) **L:** (i == 2) **R:** (i == 8) **Interval**: Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #36** : - -L - P - T - R - - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 4) **T:** (i == 6) **L:** (i == 2) **R:** (i == 8) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid
Test #37 : - -L - T - R - - - - P **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 6) **T:** (i == 6) **L:** (i == 2) **R:** (i == 8) **Interval**: Built **Expected Result**: Valid **Actual Result**:	**Test #38** : - -L - R - - P- - - - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) || (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 7) **T:** (i > i) **L:** (i == 2) **R:** (i == 4) **Interval**: Built **Expected Result**: Valid **Actual Result**: Valid

Test #39 : - L P T R - - L - - P - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) ‖ (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 2 || i == 10) **T:** (i == 3) **L:** (i == 1 || i == 7) **R:** (i == 4) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid	**Test #40** : - L P T R - - L P - T - **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) ‖ (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 2 || i == 8) **T:** (i == 3 || i == 10) **L:** (i == 1 || i == 7) **R:** (i == 4) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid
Test #41 : - L P T R - - L P – T R **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) ‖ (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 2 || i == 8) **T:** (i == 3 || i == 10) **L:** (i == 1 || i == 7) **R:** (i == 4 || i == 11) **Interval**: Built … Built **Expected Result**: Valid **Actual Result**: Valid	**Test #42** : - L P T R - - L - - P R **Pattern** : Response: (T) Responds to (P) **Scope** : After L Until R **Formula**: □((L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *W* R) **Formula:** □(□(L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) ‖ (L ∧ ¬R →(P →(¬R U (T ∧ ¬R)))) *U* R)) **P:** (i == 2 || i == 10) **T:** (i == 3) **L:** (i == 1 || i == 7) **R:** (i == 4 || i == 11) **Interval**: Built … Built **Expected Result**: Not valid **Actual Result**: Not valid

Appendix H
Verification of Trials

Property A1: List

	Global	Before R	After L	Between L and R	After L until R
Universality	2	1			
Absence				1	**C**
Existence		2			
Precedence	15	6	1		1
Response	1	1	1		1

Property A2: Braking

	Global	Before R	After L	Between L and R	After L until R
Universality			1		1
Absence					
Existence			2		
Precedence			1		
Response	13 **(C)**		10	3	

Property A3: Avionics

	Global	Before R	After L	Between L and R	After L until R
Universality			3		
Absence		2			
Existence					1
Precedence	11 **(C)**		6	1	
Response	1	1	2		

Property A4: Wheel

	Global	Before R	After L	Between L and R	After L until R
Universality			2		**(C)**
Absence					
Existence			2		
Precedence	1		1	1	1
Response	3		5	6	8

Property A5: Train

	Global	Before R	After L	Between L and R	After L until R
Universality	1			14	3 **(C)**
Absence				3	3

Existence	1	1		1	
Precedence				1	4
Response	1				

Property B1: Performance

	Global	**Before R**	**After L**	**Between L and R**	**After L until R**
Universality	1		2		1
Absence					
Existence	1		2		1
Precedence					1
Response	14 **(C)**	2	2	5	1

Property B2: ATM

	Global	**Before R**	**After L**	**Between L and R**	**After L until R**
Universality					1
Absence					
Existence			1	2	1
Precedence	1		1	3	3 **(C)**
Response	4	2	2	2	3

Property B3: BIOS

	Global	**Before R**	**After L**	**Between L and R**	**After L until R**
Universality	2	1	7		
Absence		1	9 **(C)**		
Existence		1			
Precedence	1				
Response	5		2	1	

Property B4: Process

	Global	**Before R**	**After L**	**Between L and R**	**After L until R**
Universality	1	1	2	2	
Absence					**(C)**
Existence		2	1	1	1
Precedence	9	5		1	1
Response	1		1		

Property B5: Paging

	Global	**Before R**	**After L**	**Between L and R**	**After L until R**
Universality	1		1	5	2 **(C)**

Absence					
Existence	1		1	4	1
Precedence					
Response	10		2		1

Appendix I
List of counter examples to refute subjects' responses

The "# of subject" field shows the number of subjects that made such a choise.

Property	A1
Pattern/Scope	Precedence/global
# of subjects*	15
Correct Pattern/Scope	Absence/AfterL-UntilR
# of subjects	0
Counter Example using SPIN **Test**: ~LE ~LE ~LE LE LE LE ~LE ~LE ~LE ~LD ~LD ~LD ~LD ~LD ~LD ~LD LD ~LD **Anticipated Result**: Violation. **Actual Result by SPIN**: No Violation.	

Property	A5
Pattern/Scope	Universality/Between L and R
# of subjects*	14
Correct Pattern/Scope	Universality/After L until R
# of subjects	3
Counter Example using SPIN **Test**: -- -- -- MB -- -- ~CG -- -- -- SD **Anticipated Result**: Violation. **Actual Result by SPIN**: No Violation.	

Property	B3
Pattern/Scope	Response/Global
# of subjects*	5
Correct Pattern/Scope	Absence/AfterL
# of subjects	9
Counter Example using SPIN **Test**: -- -- -- LB -- ~NS -- NS IS **Anticipated Result**: Violation. **Actual Result by SPIN**: No Violation.	

Property	B4
Pattern/Scope	Presedence/Global
# of subjects*	9
Correct Pattern/Scope	Absence/After-L Until-R
# of subjects	0
Counter Example using SPIN **Test**: RL PT ~RL PT **Anticipated Result**: Violation. **Actual Result by SPIN**: No Violation.	

Property	B5
Pattern/Scope	Response/Global
# of subjects*	10
Correct Pattern/Scope	Universality/Between-L and R
# of subjects	5
Counter Example using SPIN All 10 subjects defined P = 1MB < system memory < 2MB and T = program executed. **Test**: P P P P ~P ~T ~T ~T ~T T **Anticipated Result**: Violation. **Actual Result by SPIN**: No Violation.	