Testing the Consistency Of
Process Specifications
by
Christofis Christofi
Supervisor: M. Grüninger
March 2008
Testing the Consistency Of
Process Specifications

by
Christofis Christofi

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF APPLIED SCIENCE

DEPARTMENT OF MECHANICAL AND INDUSTRIAL ENGINEERING
UNIVERSITY OF TORONTO

Supervisor: M. Grüninger

March 2008
ABSTRACT

Businesses invest a significant amount of time and money on the development of their operating processes and rules. Far too often however these processes contain detrimental inconsistencies that the organization cannot effectively detect. The purpose of this thesis project is to use automated reasoning to experimentally test for inconsistencies in process specifications. To achieve this objective the Process Specification Language (PSL) ontology is used to provide a formal representation of the process in question, the constraints, and the queries used to support the hypotheses. The experimental results of this project support the hypotheses that through the inclusion of lemmas and reduction in background ontology size, assistance is provided to the theorem prover to maximize its reasoning efficiency. The thesis also outlines the discovery of incompleteness in the PSL ontology and provides the research community with a corrected proof of the Transitivity axiom.
ACKNOWLEDGEMENTS

I would like to thank Professor Michael Grüninger for his support and guidance over the course of this thesis project.
# TABLE OF CONTENTS

ABSTRACT .............................................................................................................................. ii
ACKNOWLEDGEMENTS ........................................................................................................ iii
LIST OF FIGURES .................................................................................................................. v
LIST OF TABLES ................................................................................................................... vi

1.0 INTRODUCTION .............................................................................................................. 1
  1.1 Purpose Statement ....................................................................................................... 1
  1.2 Motivation ................................................................................................................... 1

2.0 BACKGROUND AND OBJECTIVES OF THE PROJECT ............................................. 3

3.0 LITERATURE REVIEW ................................................................................................... 6
  3.1 The Process Specification Language Ontology .......................................................... 6
  3.2 Other Process Ontologies ............................................................................................ 9

4.0 METHODOLOGY ........................................................................................................... 10
  4.1 Overview of the Methodology .................................................................................. 10
  4.2 Background Ontologies/Theories ............................................................................. 12
  4.3 Process Description ................................................................................................... 14
    4.3.1 Complex Activity ............................................................................................... 15
    4.3.2 Precondition Axioms ......................................................................................... 18
    4.3.3 Effect Axioms .................................................................................................... 19
    4.3.4 Closure Axioms ................................................................................................. 21
  4.4 Queries ...................................................................................................................... 22
    4.4.1 Ordering Queries ............................................................................................... 22
    4.4.2 Temporal Projections Queries ............................................................................ 31
    4.4.3 Preconditions and Effects Queries ..................................................................... 37

5.0 EXPERIMENTAL RESULTS.......................................................................................... 41
  Experiments 5.1 .............................................................................................................. 41
  Experiments 5.2 .............................................................................................................. 60
  Experiments 5.3 .............................................................................................................. 79

6.0 ONTOLOGY ANALYSIS ............................................................................................. 103

7.0 SUMMARY OF EXPERIMENTS ................................................................................. 105

8.0 CONCLUSION ............................................................................................................... 108

9.0 REFERENCES ............................................................................................................... 109

APPENDICES ...................................................................................................................... 110
  Appendix A: Drain Process Modeling .......................................................................... 111
  Appendix B: Otter Proofs Generated Through Experiments .......................................... 114
  Appendix C: Otter Proof of Lemma 1 .......................................................................... 119
LIST OF FIGURES

Figure 2.1. Steps in the Sanitize Process ................................................................. 4
Figure 2.2. Steps in the Drain Process ..................................................................... 5
Figure 4.3.1.1. Process Description – Sanitize Process (Otter Syntax) ..................... 15
Figure 4.3.1.2. Subactivity Relationships – Sanitize Process (Otter Syntax) ............. 17
Figure 4.3.1.3. Subactivity Tree for Sanitize and Drain ........................................... 18
Figure 4.3.2.1. Precondition Axioms – Sanitize Process (Otter Syntax) .................. 18
Figure 4.3.3.1. Effect Axioms – Sanitize Process (Otter Syntax) ............................ 20
Figure 4.3.4.1. Process Description with Closure – Sanitize Process (Otter Syntax) .... 21
Figure 6.1. Manual proof of the Transitivity Axiom ................................................ 104
LIST OF TABLES

Table 3.1. Lexicon of the PSL Ontology utilized ................................................................. 8
Table 3.2. Alternative Approaches as compared to PSL ...................................................... 9
Table 5.1. Outline of Experiments 5.1 ................................................................................ 41
Table 5.2. Outline of Experiments 5.2 ................................................................................ 60
Table 5.3. Outline of Experiments 5.3 ................................................................................ 79
Table 7.1. Comparison of Experiments which generated proofs ....................................... 106
1.0 INTRODUCTION

1.1 Purpose Statement

The purpose of this thesis project is to apply automated reasoning to experimentally test for inconsistencies in process specifications and associated constraints. To achieve this objective the Process Specification Language (PSL) ontology is utilized to provide a formal representation of the process description, the constraints, and the queries used in the experimental analysis. Two hypotheses, one relating to the reduction of the ontology size, the other relating to the inclusion of lemmas to assist the theorem prover, will be the focus of this investigation.

1.2 Motivation

Today businesses invest a lot of money in the development of their business and manufacturing processes. To some extent, the success of a company can be determined by the quality of the processes that govern the business function. In manufacturing, processes drive the company’s outputs – the quality of them often determines the reputation and success of the business. The importance of processes within industry is unparalleled. Far too often these processes contain inconsistencies that may be undetectable at the onset of implementation. These inconsistencies can lead to process problems and result in consequences detrimental to the success of the organization. Such consequences include the inability to meet customer demands, inefficient operation of processes, higher costs, exposure to risks, and the lack of a competitive edge in industry.
There exists a need to represent manufacturing process knowledge in an unambiguous manner to facilitate the ability to automatically detect inconsistencies in process specifications. The Process Specification Language (PSL) ontology is a standardized approach to the concepts of process executions. PSL has previously been applied successfully in scheduling, process modeling semantics, process planning, production planning, simulation, project management, workflow, and business process reengineering (NIST). The formalism in the PSL representation supports automated reasoning and coupled with a theorem proving software can be used to effectively detect inconsistencies in process specifications.
2.0 BACKGROUND AND OBJECTIVES OF THE PROJECT

The objective of this project is to apply automated reasoning to experimentally test for inconsistencies in process specifications. The approach to achieving this objective will be experimental. Initially a suitable process description is acquired and formally modeled using the PSL ontology. The modeling of the process description using PSL and the generation of the queries is discussed in Section 4.0 of the report. The following is a description of the selected process used throughout the thesis project.

The “Hamilton Beach 4200 Series Water Dispenser” was identified as a concrete example with which it is possible to extract process specifications similar to those in a real-life manufacturing setting. The specific focus will be on the Sanitize process extracted from the use and care manual that was created for this device. This process is appropriate for purposes of representation because it contains many aspects and constraints typical of a manufacturing maintenance process utilized in industry. By identifying a tangible example, such as the sanitization process of a water dispenser, on a smaller scale, the findings can be scaled up and applied to larger applications.

The Hamilton Beach Dispenser Sanitize process consists of a set of 18 steps, both primitive and complex. To differentiate, a primitive activity is one which does not have any subactivities (i.e. cannot be broken down into smaller parts), and a complex activity is one that does have subactivities (i.e. can be broken down into smaller parts). To demonstrate, consider the following list of steps that composes the Sanitize process:
To Sanitize:

**Important:** Before cleaning, set both power switches (on the back of the dispenser) to the OFF position and **unplug the water dispenser.** **Note:** For first time installation, skip next three steps.

1) Empty the water bottle, then remove it from the dispenser.
2) Drain the water tanks (see "Draining Instruction")
3) Reinstall the silicon stoppers and drain clips, (see "Draining Instruction")
4) **DO NOT ADD BLEACH FIRST OR DIRECTLY TO THE DISPENSER.**
   Concentrated bleach may damage plastic. Add 1/2 teaspoon (2.5 ml) of ordinary 6.0% maximum household bleach to a 2 quart (2 liter) pitcher filled with tap water.
5) Pour the solution in the opening at the top of the dispenser.
6) Repeat last two steps until the dispenser is full.
7) Allow to sit for 15-20 minutes.
8) **Rinse out the bleach solution.**
9) Drain the water tanks into a large bucket (see "Draining Instruction").
10) Make sure that the silicon stoppers are **NOT** installed.
11) Clean, then reinstall the collar
12) Pour approximately 1 gallon (4 quarts or 4 liters) of fresh tap water into the top and allow to drain into the bucket.
13) Flush with an additional 1 gallon (4 quarts or 4 liters) of fresh tap water and allow to drain into the bucket.
14) **Reinstall the silicon stoppers and drain clips** (see "Draining Instruction").
15) Install a fresh water bottle (see "Preparing and Loading Your Dispenser for Use").
16) Plug in the dispenser. Then turn on the hot and cold power switches (on the back of the dispenser).
17) Dispense 1 quart (1 liter) of water from faucet and taste to make sure the bleach solution is completely removed.
18) If you continue to taste the chlorine from the bleach solution, **repeat steps 10 - 18** until the taste is gone.

The first step, “Empty the water bottle…”, is a primitive activity, it has no subactivities. The second step, “Drain the water tanks…”, is a complex activity, as it refers to another process of “Draining Instructions” which describes its subactivities. The “Draining Instructions” will also need to be represented to support the complex activities present in the Sanitize process.

Below are the steps that compose the Drain process:
Figure 2.2. Steps in the Drain Process

**Draining Instruction:**

The dispenser should be drained before cleaning the tanks and before leaving for long vacations or absences.

1) Empty all water in bottle by dispensing cold water.
2) Set both power switches (on the back of the dispenser) to the **OFF** position and unplug the dispenser.

   **WARNING: BURN HAZARD. ALLOW THE DISPENSER TO SIT FOR 30 MINUTES TO ALLOW HOT WATER TO COOL BEFORE PROCEEDING**

   The dispenser heats water to a temperature of approximately 190 °F (88 °C). Water temperatures above 125 °F (52 °C) can cause severe burns or death from scalding. Children, the disabled and the elderly are at highest risk of being scalded. Children should be supervised by an adult when using this product.

3) Remove the water bottle from the dispenser.
4) Place the dispenser over a kitchen sink or basin that can withstand hot water beneath the drain outlets of the dispenser.
5) Remove the clip and silicon stopper from the drain outlets and allow water to drain.
6) After draining, replace the silicon stopper and clip.
7) Make sure to install a full bottle before turning on the dispenser.

Utilizing the PSL ontology to fully represent the Sanitize and Drain process specifications allows for an unambiguous representation that can be automatically tested for consistency with the help of theorem proving software.
3.0 LITERATURE REVIEW

This section of the report describes the Process Specification Language Ontology and its function, introduces first order logic, and describes other approaches to process ontologies.

3.1 The Process Specification Language Ontology

The Process Specification Language ontology was created to facilitate correct and complete exchange of process information among manufacturing systems (Grüninger and Menzel 63). It outlines an unbiased approach to representing processes in a manufacturing setting and, in so doing, supports automated reasoning.

Consider the manufacturing world today and the multitude of software applications aiding in manufacturing operations. Typically interoperability among these various applications is obstructed because they use different terminology and representations of the domain (Grüninger and Menzel 63). In practice, point-to-point translation programs are often constructed to facilitate this communication between applications (Grüninger and Menzel 63). A problem that often arises, even with the translation programs enabling applications to use the same terminology, is that the applications often assign different semantics with terms thus preventing a complete exchange of information among the applications. Moreover the number of applications is increasing, in turn increasing the number and complexity of translation programs to be created to accommodate this.
PSL aims to provide a process representation that is common to all manufacturing applications, flexible enough to be applied to any application and robust enough to represent the necessary process information (NIST).

The PSL ontology, using first-order logic (FOL) theories, defines a set of core theories (PSL-Core) and sets of definitional extensions that build upon these core theories (Grüninger 4).

The FOL domain consists of Objects (e.g. cat, dog, ball, house, etc.), Relations (e.g. red, round, big, etc.), and Functions (e.g. father_of, best_friend, etc.). The lexicon consists of logical symbols and nonlogical symbols. Logical symbols include parenthesis, logical connectives (, , , , , , ), variables (x, y, etc.), equality (=), and quantifiers ( , ). Nonlogical symbols include predicate symbols (e.g. P(…)) and function symbols (e.g. P, beginof, endof, etc.). The lexical elements in FOL, as with any other language, are governed by grammar and syntax. An incredibly powerful aspect of FOL is the concept of quantifiers. Quantification is important because it allows you to express the extent to which a predicate is true over a range of elements. There are two types of quantifiers, universal ( ) and existential ( ). Universal quantifiers express that a predicate is true for all elements under consideration. Existential quantifiers express that there is one or more elements under consideration for which the predicate is true. Using this PSL is able to define its theories.

The PSL-Core theories act as the basic units of translation for the fundamental concepts of manufacturing processes. The extensions act to support the core theories by introducing new terminology capable of providing definitions for additional manufacturing processes not
described by the core (Grüninger and Menzel 64). The Core ontology consists of four classes, each of which is described below (Grüninger 5, Grüninger and Menzel 64):

i. *Activities* – a repeatable pattern of behavior that may have zero or more occurrences.

ii. *Activity Occurrences* – corresponds to a concrete instantiation of an activity. Activity occurrences must begin and end at timepoints.

iii. *Timepoints* – a linearly ordered set with endpoints at infinity.

iv. *Objects* – those elements that are not activities, occurrences, or timepoints.

The following chart outlines the lexicon of core theories to be utilized in this project (Grüninger 4).

**Table 3.1.** Lexicon of the PSL Ontology utilized

| T_{pslcore} | activity(a) | a is an activity |
|            | activity_occurrence(o) | o is an activity occurrence |
|            | object(x) | x is an object |
|            | occurrence_of(o,a) | o is an occurrence of a |
| T_{subactivity} | subactivity(a_1,a_2) | a_1 is a subactivity of a_2 |
|            | primitive(a) | a is a minimal element of the subactivity ordering |
| T_{occtree} | legal(s) | s is an element of a legal occurrence |
|            | earlier(s_1,s_2) | s_1 precedes s_2 in an occurrence tree |
| T_{disc_state} | holds(f,s) | the fluent f is true immediately after the activity occurrence s |
|            | prior(f,s) | the fluent f is true immediately before the activity occurrence s |
| T_{complex} | min_precedes(s_1,s_2,a) | The atomic subactivity occurrence s_1 precedes the atomic subactivity s_2 in an activity tree for a |
| T_{actocc} | root_occ(o) | the initial atomic subactivity occurrence of o |
|            | leaf_occ(s,o) | s is the final atomic subactivity occurrence of o |
3.2 Other Process Ontologies

The PSL methodology can be compared to some other approaches. The data in the following table describes some alternative approaches and their limitations, as compared to PSL (Bock and Grüninger 229-230).

Table 3.2. Alternative Approaches as compared to PSL

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Description</th>
<th>Comparison to PSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Petri nets</td>
<td>- Designed to model the synchronization of concurrent processes</td>
<td>- no standard semantics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- advanced knowledge of complex mathematics required for understanding</td>
</tr>
<tr>
<td>Planning Domain Definition Language (PDDL)</td>
<td>- spans the domain of planning, including a specification of states, the set of possible activities, the structure of complex activities, and the effects of activities</td>
<td>- does not provide the first-order logic expressions of the items mapped to mathematical structures</td>
</tr>
<tr>
<td>GOLog</td>
<td>- provides mechanisms for specifying complex activities as programs in a second order language that extends the axiomatization of situation calculus</td>
<td>- describes its mathematical structures informally</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- does not prove equivalence to its axiomization</td>
</tr>
<tr>
<td>Workflow Process Definition Language (WPDL)</td>
<td>- standard terminology which can serve as a common framework for different workflow management systems</td>
<td>- does not provide a mapping to mathematical structures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- does not provide a semantic domain</td>
</tr>
<tr>
<td>DARPA Agent Markup Language (DAML-S)</td>
<td>- provides a set of process classes that can be specialized to describe a variety of Web services</td>
<td>- does not provide a mapping to mathematical structures</td>
</tr>
<tr>
<td>Unified Modeling Language (UML)</td>
<td>- describe how objects change over time using a series of object snapshots</td>
<td>- does not validate semantic domains with mathematical structures</td>
</tr>
<tr>
<td></td>
<td>- modeled as constraints on object snapshots before and after execution of the behavior</td>
<td>- does not axiomatize domains in first-order logic</td>
</tr>
</tbody>
</table>
4.0 METHODOLOGY

The following section provides an overview of the steps that will be taken to accomplish the objectives of the project. The methodology includes identifying an appropriate process for representation, identifying the background theories to be used in the reasoning process, creating and introducing the specific axioms necessary for representation (i.e. the process description), and generating and testing queries in an experimental fashion.

4.1 Overview of the Methodology

To achieve the objective of automatically detecting inconsistencies in process specifications an experimental approach was followed. Initially a suitable process description is acquired and formally modeled using the PSL ontology. Two versions of the process specifications are utilized in the experiments. With this in hand, a set of queries focusing on the ordering relations is created and used as the basis of the experiments. Queries surrounding temporal projections and preconditions and effects were also created but not implemented for any experimentation. There are two main types of ordering queries used, namely retrieval queries, and verification queries. In addition to these queries a set of background theories are identified and used as variables to test the hypotheses of the experiments. A total of five theories are used and vary in regards to the number of axioms and definitions they contain. Three of the background theories are identified as lemmas and are included in different sets of experiments.

The first family of hypotheses that are to be investigated surround the discovery and inclusion of lemmas to improve the performance of the theorem prover. These lemmas are
provided to Otter to improve the speed at which the system can generate a proof. By providing the theorem prover with axioms that it would have in fact proven along the course of its reasoning, valuable time can be saved and the theorem prover can focus on generating a proof of the query instead of complicated background axioms needed for completing the proof.

The second family of hypotheses that are to be investigated surround the size of the background ontology. By reducing the number of axioms the theorem prover must manage, valuable computation time can be conserved and more time can be spent developing a proof for the query versus drudging through potentially unnecessary axioms. The three choices of background ontology include $T_{\text{actoc}} \cup \text{complex}$, $T_{\text{complex}}$, and no additional background theory at all (i.e. the background ontology consists of only the process description and the lemmas provided).
4.2 Background Ontologies/Theories

A number of experiments were executed, each incorporating a different theory with the queries generated (described in Section 4.4). As mentioned in Section 4.1, these background theories are used in the experiments in addition to one of the two process descriptions created. The two theories and three lemmas that were used include $T_{\text{actocc}} \cup \text{complex}$, $T_{\text{complex}}$, $T_{\text{lemma}_1}$, $T_{\text{lemma}_2}$, and $T_{\text{lemma}_3}$.

$T_{\text{actocc}} \cup \text{complex}$ consists of two core theories, namely $T_{\text{actocc}}$ and $T_{\text{complex}}$. $T_{\text{actocc}}$ consists of 18 axioms and 5 definitions. These axioms help to ensure that all complex activity occurrences correspond to branches of activity trees (NIST). $T_{\text{complex}}$ consists of 12 axioms and 5 definitions. This theory enables the ability to represent and reason about complex activities and the relationship between occurrences of an activity and occurrences of its subactivities (NIST).

The purpose of including the lemmas as additional theories is to support the hypothesis that by providing the theorem prover with axioms that it would derive along the course of its reasoning, it will reduce the computational time it takes to generate a proof. These lemmas were chosen because they are used in the manual proofs of the queries. The three lemmas are described below:

$T_{\text{lemma}_1}$ includes the single axiom:

\[
\forall s \, \forall a \, (\text{root}(s, a) \rightarrow (\exists o \, (\text{occurrence_of}(o, a) \land \text{subactivity_occurrence}(s, o)))).
\]
\( T_{\text{lemma}_2} \) includes, in addition to the axiom present in \( T_{\text{lemma}_1} \), the Transitivity axiom:

\[
\begin{align*}
\text{all } s_1 & \text{ all } s_2 & \text{ all } s_3 & \text{ all } a \ ( \\
& (\text{min_precedes}(s_1,s_2,a) & \text{ & } \text{min_precedes}(s_2,s_3,a)) \\
\rightarrow & \text{min_precedes}(s_1,s_3,a)) .
\end{align*}
\]

\( T_{\text{lemma}_3} \) includes, in addition to the axioms present in \( T_{\text{lemma}_2} \), the axiom:

\[
\begin{align*}
\text{all } s_1 & \text{ all } s_2 & \text{ all } s_3 & \text{ all } a \ ( \\
& (\text{min_precedes}(s_1,s_2,a) & \text{ & } \text{min_precedes}(s_3,s_2,a)) \\
\rightarrow & (\text{min_precedes}(s_1,s_3,a) \text{ } | \text{ } \text{min_precedes}(s_3,s_1,a) \text{ } | \\
& (s_1=s_3))).
\end{align*}
\]

The relationship among the five background theories is summarized below:

\[
\begin{array}{c|c}
T_{\text{actocc} \cup \text{complex}} & T_{\text{complex}} \\
T_{\text{complex}} & T_{\text{lemma}_1} \\
T_{\text{complex}} & T_{\text{lemma}_2} \\
T_{\text{complex}} & T_{\text{lemma}_3}
\end{array}
\]

As is evident from their composition, the background theories vary in size (i.e. the largest being \( T_{\text{actocc} \cup \text{complex}} \) and the smallest being \( T_{\text{lemma}_1} \)). The idea here is to discover the minimal set of axioms (i.e. the weakest theory) that will enable the theorem prover to generate a proof. This supports the second hypothesis surrounding background ontology sizes. As mentioned in Section 4.1, by reducing the number of axioms the theorem prover must manage, ideally more time can be spent focusing the efforts of the theorem prover on the query at hand.
4.3 Process Description

The “Hamilton Beach 4200 Series” water dispenser was identified as an appropriate candidate for representation. The Sanitize process will be the focus of the representation. The water dispenser process specifications contain similar steps and constraints that would typically be present in a large-scale manufacturing process. By successfully modeling a smaller scale example it is hoped that it will provide a basis for larger applications.

The Process Description acts as the PSL representation of the “English” instructions described in Section 2.0 of the report. The process description identifies under what conditions activities occur, describes how occurrences of activities change fluents (or states of the world), and preserves the ordering of the process steps. The general form of a process description in PSL representation is:

\[(\forall o) \text{occurrence}_o(A) \supset (\exists o_n) \text{occurrence}_o(o_n, \Phi)\]

The above statement says “For all occurrences of activity A, there exists occurrences of activities \(\Phi\)”, where \(\Phi\) is a number of different activities occurring when activity A is initiated and \(n\) is the number of different occurrences. Combined with effect axioms and precondition axioms the process is fully represented.
4.3.1 Complex Activity

Below is the process description created for the Sanitize process. As the Otter theorem prover will be utilized in testing for inconsistencies all representations are presented using Otter syntax. This version of the process description was used in experiments 5.1.1-5.1.36, and is denoted $\sum_{pd\_complex}$ in the experiment summaries of Section 5.0.

Figure 4.3.1.1. Process Description – Sanitize Process (Otter Syntax)

```
all o (occurrence_of(o,sanitize(Dispenser)) ->
(exists o1 exists o2 exists o3 exists o4 exists o5 exists o6
exists o7 exists o8 exists o9 exists o10 exists o11 exists o12
exists o13 exists o14 exists o15 exists o16 exists o17
exists o31 exists o91 exists o141 exists o151
  (occurrence_of(o1,empty(Bottle))
& occurrence_of(o2,removebottle(Bottle,Dispenser))
& occurrence_of(o3,drain(Water,Dispenser))
& occurrence_of(o4,reinstall(Siliconstopper,Drainclip,Dispenser))
& occurrence_of(o5,add(Solution,Pitcher))
& occurrence_of(o6,pour(Solution,Dispenser))
& occurrence_of(o7,sit(Solution,Dispenser))
& occurrence_of(o8,rinse(Solution,Dispenser))
& occurrence_of(o9,drain(Water,Dispenser))
& occurrence_of(o10,clean(Collar))
& occurrence_of(o11,reinstallcollar(Collar,Dispenser))
& occurrence_of(o12,pour(Water,Dispenser))
& occurrence_of(o13,flush(Water,Dispenser))
& occurrence_of(o14,reinstall(Siliconstopper,Drainclip,Dispenser))
& occurrence_of(o15,install(Bottle,Dispenser))
& occurrence_of(o16,plug(Dispenser))
& occurrence_of(o17,dispense(Water,Dispenser))
& min_precedes(o1,o2,sanitize(Dispenser))
& min_precedes(o2,root_occ(o3),sanitize(Dispenser))
& min_precedes(o31,o4,sanitize(Dispenser))
& leaf_occ(o31,o3)
& min_precedes(o4,o5,sanitize(Dispenser))
& min_precedes(o5,o6,sanitize(Dispenser))
& min_precedes(o6,o7,sanitize(Dispenser))
& min_precedes(o7,o8,sanitize(Dispenser))
& min_precedes(o8,root_occ(o9),sanitize(Dispenser))
& min_precedes(o91,o10,sanitize(Dispenser))
& leaf_occ(o91,o9)
& min_precedes(o10,o11,sanitize(Dispenser))
& min_precedes(o11,o12,sanitize(Dispenser))
& min_precedes(o12,o13,sanitize(Dispenser))
& min_precedes(o13,root_occ(o14),sanitize(Dispenser))
& min_precedes(o141,root_occ(o15),sanitize(Dispenser))
```

15
As mentioned in Section 4.3 the representation preserves the ordering relationship of the steps in the process. This is accomplished by the use of the min_precedes(s1, s2, a) relation. This relation implies “the atomic subactivity occurrence s1 precedes the atomic subactivity s2 in an activity a”. As an example from Figure 4.3.1.1, consider the following statement:

\[
\text{min_precedes(o1, o2, sanitize(Dispenser))}
\]

This statement tells us that o1 (an occurrence of the empty activity) precedes o2 (an occurrence of the remove activity) in the sanitize process. From Figure 2.1, step 1, one can see that this statement says “you must empty the water bottle before removing it”. The preservation of ordering acts as a constraint within the representation that specifies that one activity occurrence must be completed before another activity occurrence can begin. For complex activities the root_occ(o)(which specifies the initial atomic subactivity occurrence of o), and leaf_occ(s, o)(which says s is the final atomic subactivity
occurrence of o) relations are utilized. The best method of describing the use of these relations is by example. Consider the following statements from the process specification in Figure 4.3.1.1:

\[
\text{min\_precedes(o2, root\_occ(o3), sanitize(Dispenser))}
\]
\[
\text{^min\_precedes(o3', o4, sanitize(Dispenser))}
\]
\[
\text{^leaf\_occ(o3', o3)}
\]

The first statement says that o2 precedes the initial subactivity of o3 (which is an occurrence of drain, a complex activity) in the sanitize process. The next statement says that o3′ precedes o4 (which is an occurrence of reinstall) in the sanitize process. The final statement defines o3′ as the final subactivity of the activity occurrence o3. To define the subactivity relationships in the process description the following statements are included:

**Figure 4.3.1.2. Subactivity Relationships – Sanitize Process (Otter Syntax)**

- subactivity(empty(Bottle), sanitize(Dispenser)).
- subactivity(removebottle(Bottle, Dispenser), sanitize(Dispenser)).
- subactivity(drain(Water, Dispenser), sanitize(Dispenser)).
- subactivity(reinstall(Siliconstopper, Drainclip, Dispenser), sanitize(Dispenser)).
- subactivity(add(Solution, Pitcher), sanitize(Dispenser)).
- subactivity(pour(Solution, Dispenser), sanitize(Dispenser)).
- subactivity(sit(Solution, Dispenser), sanitize(Dispenser)).
- subactivity(rinse(Solution, Dispenser), sanitize(Dispenser)).
- subactivity(clean(Collar), sanitize(Dispenser)).
- subactivity(reinstallcollar(Collar, Dispenser), sanitize(Dispenser)).
- subactivity(flush(Water, Dispenser), sanitize(Dispenser)).
- subactivity(install(Bottle, Dispenser), sanitize(Dispenser)).
- subactivity(plug(Dispenser), sanitize(Dispenser)).
- subactivity(dispense(Water, Dispenser), sanitize(Dispenser)).

Each of the statements in Figure 4.3.1.2 identify the activities that are subactivities of the Sanitize process. As mentioned earlier, some of the activities in the Sanitize process are complex and refer to the Drain process. The process description of the Drain process can be found in Appendix A, Figure A1, and the subactivity relationships are in Appendix A, Figure A2. As the Sanitize process and the Drain process are somewhat linked, it is interesting to
observe subactivities common to both. Outlined below is a subactivity tree describing the subactivities of the *Sanitize* and *Drain* processes. Those subactivities common to both are indicated with an asterisk.

**Figure 4.3.1.3. Subactivity Tree for *Sanitize* and *Drain***

4.3.2 Precondition Axioms

Precondition axioms specify the conditions under which an activity can possibly occur. The general form of a precondition axiom is the following:

\[(\forall o) \ occurrence\_of(o,A) \land legal(o) \Rightarrow prior(f,o)\]

The above statement says that before an occurrence of activity A that is legal (i.e. allowed to occur under some behavior specification (Bock and Grüninger 212)) can occur, the state f must be true. A set of precondition axioms are added to the process description in order to completely describe the conditions under which activities can occur. The following are the precondition axioms for the *Sanitize* process:

**Figure 4.3.2.1. Precondition Axioms – *Sanitize* Process (Otter Syntax)**

```
all o (occurrence_of(o,remove(Bottle,Dispenser)) & legal(o) -> prior(bottleempty(Bottle),o) ).
all o (occurrence_of(o,drain(Water,Dispenser)) & legal(o) -> -prior(installedclips(Siliconstopper,Drainclip,Dispenser),o) & prior(bottleoff(Bottle,Dispenser),o) ).
all o (occurrence_of(o,reinstall(Siliconstopper,Drainclip,Dispenser)) & legal(o) -> prior(tankempty(Dispenser),o) ).
all o (occurrence_of(o,add(Solution,Pitcher)) & legal(o) -> prior(installedclips(Siliconstopper,Drainclip,Dispenser),o) ).
```
all o (occurrence_of(o,pour(Solution,Dispenser)) & legal(o) -> prior(mixed(Solution),o) ).
all o (occurrence_of(o,sit(Solution,Dispenser)) & legal(o) -> prior(filled(Dispenser),o) ).
all o (occurrence_of(o,rinse(Bleach,Dispenser)) & legal(o) -> prior(bleachpresent(Dispenser),o) ).
all o (occurrence_of(o,drain(Water,Dispenser)) & legal(o) -> prior(bleachpresent(Dispenser),o) ).
all o (occurrence_of(o,clean(Collar)) & legal(o) -> prior(collarinstalled(Collar,Dispenser),o) & prior(installedclips(Siliconstopper,Drainclip,Dispenser),o) )
all o (occurrence_of(o,pour(Water,Dispenser)) & legal(o) -> prior(cleaned(Collar),o) & prior(collarinstalled(Collar,Dispenser),o))
all o (occurrence_of(o,flush(Water,Dispenser)) & legal(o) -> prior(cleaned(Collar),o) & prior(collarinstalled(Collar,Dispenser),o))
all o (occurrence_of(o,reinstall(Siliconstopper,Drainclip,Dispenser)) & legal(o) -> prior(cleaned(Dispenser),o) & prior(tankempty(Dispenser),o))
all o (occurrence_of(o,install(Bottle,Dispenser)) & legal(o) -> prior(installedclips(Siliconstopper,Drainclip,Dispenser),o))
all o (occurrence_of(o,plug(Dispenser)) & legal(o) -> prior(bottleoff(Bottle,Dispenser),o))
all o (occurrence_of(o,dispense(Water,Dispenser)) & legal(o) -> prior(poweroff(Dispenser),o))

To illustrate the use of the precondition axioms consider the first one listed in Figure 4.3.2.1. This statement describes that before an occurrence of remove can occur, the bottleempty state (i.e. the bottle is empty) must be true. Precondition axioms for the Drain process can be found in Appendix A, Figure A3.

4.3.3 Effect Axioms

Effects define the ways in which activity occurrences change the state of the world (or fluents). Effects can be context-free, so that all occurrences of the activity change the same states, or they may be constrained by other conditions (Grüninger 11). The general form of an effect axiom is the following:

\(( \forall o ) \ occurrence \_of(o,A) \supset holds(f,o)\)
The above statement says that after an activity occurrence of A, the state f will be true. A set of effect axioms are added to the process description to describe the effects of the activity occurrences in the process. The following are the effect axioms for the activity occurrences of the Sanitize process:

**Figure 4.3.3.1. Effect Axioms – Sanitize Process (Otter Syntax)**

```plaintext
all o (occurrence_of(o,empty(Bottle)) -> holds(bottleempty(Bottle),o)).
all o (occurrence_of(o,removebottle(Bottle,Dispenser)) ->
    holds(bottleoff(Bottle,Dispenser),o)).
all o (occurrence_of(o,drain(Water,Dispenser)) ->
    holds(tankempty(Dispenser),o)).
all o (occurrence_of(o,removebottle(Bottle,Dispenser)) ->
    holds(bottleoff(Bottle,Dispenser),o)).
all o (occurrence_of(o,drain(Water,Dispenser)) ->
    holds(tankempty(Dispenser),o)).
all o (occurrence_of(o,reinstall(Siliconstopper,Drainclip,Dispenser)) ->
    holds(installedclips(Siliconstopper,Drainclip,Dispenser),o)).
all o (occurrence_of(o,reinstallcollar(Collar,Dispenser)) ->
    holds(collarinstalled(Collar,Dispenser),o)).
all o (occurrence_of(o,add(Solution,Pitcher)) ->
    holds(mixed(Solution),o)).
all o (occurrence_of(o,pour(Solution,Dispenser)) ->
    holds(filled(Dispenser),o)).
all o (occurrence_of(o,sit(Solution,Dispenser)) ->
    holds(bleachedpresent(Dispenser),o)).
all o (occurrence_of(o,rinse(Bleach,Dispenser)) ->
    - holds(bleachpresent(Dispenser),o)).
all o (occurrence_of(o,clean(Collar)) -> holds(cleaned(Collar),o)).
all o (occurrence_of(o,pour(Water,Dispenser)) ->
    holds(cleaned(Dispenser),o) & holds(tankempty(Dispenser),o)).
all o (occurrence_of(o,flush(Water,Dispenser)) ->
    holds(cleaned(Dispenser),o) & holds(tankempty(Dispenser),o)).
all o (occurrence_of(o,install(Bottle,Dispenser)) ->
    holds(bottleoff(Bottle,Dispenser),o)).
all o (occurrence_of(o,plug(Dispenser)) ->
    holds(poweroff(Dispenser),o)).
```

To illustrate the use of effect axioms, consider the first one listed in Figure 4.3.3.1. This statement describes that after every occurrence of the empty activity, the state bottleempty (i.e. the bottle is empty) will be true. Effect axioms for the Drain process can be found in Appendix A, Figure A4.
4.3.4 Closure Axioms

A process description for the *Sanitize* process considering closure was created for use in the experiments as well. The purpose of this form of the process description is to counter any problems that the theorem prover might have with failing to prevent legal occurrences of other activities. Although the process description in figure 4.3.1.1 exhaustively describes the process, it allows for undesired executions of processes that the theorem prover would consider legal according to the occurrence tree. Adding these additional “closed world” constraints can become somewhat cumbersome and as a result of this, the process description considering these closure constraints has been significantly reduced. This process description is shown below in Figure 4.3.4.1. It has been used in experiments 5.2.1-5.3.45 and is denoted $\sum_{pd\_closure}$ in Section 5.0. As well, a process description considering closure has also been created for the *Drain* process and can be found in Appendix A, Figure A5.

**Figure 4.3.4.1. Process Description with Closure – *Sanitize* Process (Otter Syntax)**

all o (occurrence_of(o,sanitize(Dispenser))
-&gt;
(exists o1 exists o2 exists o3
(occurrence_of(o1,empty(Bottle))
 & occurrence_of(o2,removebottle(Bottle,Dispenser))
 & occurrence_of(o3,install(Bottle,Dispenser))
 & min_precedes(o1,o2,sanitize(Dispenser))
 & min_precedes(o2,o3,sanitize(Dispenser))
 & subactivity_occurrence(o1,o)
 & subactivity_occurrence(o2,o)
 & subactivity_occurrence(o3,o)
 & (all s \{subactivity_occurrence(s,o)
 &\(-&gt;
 ((s=o1) | (s=o2) | (s=o3)))))))).
4.4 Queries

The following section describes the queries that were utilized in the experiments outlined in Section 5.0 of the report. An English paraphrase of the query is provided to assist the reader with comprehension of the queries. There are two main types of queries, namely retrieval queries (used to retrieve information from the process description) and verification queries (used to determine consistency of the process description). In Section 4.4.1, queries surrounding the ordering of the process description are described. In Sections 4.4.2 and 4.4.3 additional queries surrounding temporal projections, and preconditions and effects are described.

4.4.1 Ordering Queries

Query 1a

In the following Otter syntax the retrieval query states:

Does there exist an activity a, where for any occurrence tree of a there exists an occurrence of unplug(Dispenser) that precedes an occurrence of turnon(Dispenser)? The query is to return the value of activity a that makes this statement true. This query was used in experiments 5.3.1, 5.3.16, and 5.3.31.

--------------------------------------------------------------------------------------------------------------
exists a (activity(a)
  & (all s (root(s,a) ->
    (exists s1 exists s2 exists s
     (occurrence_of(s1,unplug(Dispenser))
     & occurrence_of(s2,turnon(Dispenser))
     & min_precedes(s,s1,a)
     & min_precedes(s,s2,a)
     & min_precedes(s1,s2,a)
     & $ans(a)))))).
--------------------------------------------------------------------------------------------------------------
Query 1b

In the following Otter syntax the retrieval query states:

For any activity tree for sanitize(Dispenser), there exists an occurrence of unplug(Dispenser) (s1) that precedes an occurrence of turnon(Dispenser) (s2) in activity a. The query is to return the correct activity a which would make the statement true. This query was used in experiments 5.3.2, 5.3.17, and 5.3.32.

all s (root(s,sanitize(Dispenser)))

->
(exists s1 exists s2 exists s
(occurrence_of(s1,unplug(Dispenser))
& occurrence_of(s2,turnon(Dispenser))
& min_precedes(s,s1,a)
& min_precedes(s,s2,a)
& min_precedes(s1,s2,a)
& $ans(a)))
.

--------------------------------------------------------------------------------------------------------------

Query 1c

In the following Otter syntax the retrieval query states:

Does there exists an activity a, such that for all occurrences of a there exists an occurrence of unplug(Dispenser) (s1) that precedes an occurrence of turnon(Dispenser) (s2)? This query is to return the activity a that would make the statement true. This query was used in experiments 5.3.3, 5.3.18, and 5.3.33.

exists a (activity(a)

& (all o (occurrence_of(o,a)

->
(exists s1 exists s2 exists s
(occurrence_of(s1,unplug(Dispenser))
& occurrence_of(s2,turnon(Dispenser))
& min_precedes(s,s1,a)
& min_precedes(s,s2,a)
& min_precedes(s1,s2,a)
& $ans(a)))
))
.

--------------------------------------------------------------------------------------------------------------
Query 2a

In the following Otter syntax the retrieval query states:

Does there exist an activity a, such that for any activity tree for a where there is an occurrence of unplug(Dispenser) (s1) and an occurrence of turnon(Dispenser) (s2), that s1 precedes s2? The query is to return the activity a that satisfies this statement. This query was used in experiments 5.1.1, 5.1.13, 5.1.25, 5.2.1, 5.2.13, 5.2.25, 5.3.4, 5.3.19, and 5.3.34.

```
exists a (activity(a)
    & (all s all s1 all s2
        ( ( root(s,a)
            & occurrence_of(s1,unplug(Dispenser))
            & occurrence_of(s2,turnon(Dispenser))
            & min_precedes(s,s1,a)
            & min_precedes(s,s2,a))
        ->
            (min_precedes(s1,s2,a)))).
```

Query 2b

In the following Otter syntax the verification query states:

For any occurrence tree for sanitize(Dispenser) and all occurrences of a1 (s1) and a2 (s2), s1 precedes s2. This query was used in experiments 5.1.2, 5.1.14, 5.1.26, 5.2.2, 5.2.14, 5.2.26, 5.3.5, 5.3.20, and 5.3.35.

```
all s all s1 all s2 all a1 all a2
(( root(s,sanitize(Dispenser))
    & occurrence_of(s1,a1)
    & occurrence_of(s2,a2)
    & min_precedes(s,s1,sanitize(Dispenser))
    & min_precedes(s,s2,sanitize(Dispenser)))
->
    (min_precedes(s1,s2,sanitize(Dispenser)))).
```
Query 2c

In the following Otter syntax the retrieval query states:

Does there exist an activity $a$, such that for all occurrences of $a$, and all subactivity occurrences of $a$ (unplug(Dispenser) ($s_1$) and turnon(Dispenser) ($s_2$)) $s_1$ precedes $s_2$? The query is to return the activity $a$ that satisfies this statement. This query was used in experiments 5.1.3, 5.1.15, 5.1.27, 5.2.3, 5.2.15, 5.2.27, 5.3.6, 5.3.21, and 5.3.36.

```
exists a
   (activity(a)
     & (all o all s1 all s2
        ( occurrence_of(o,a)
          & occurrence_of(s1,unplug(Dispenser))
          & occurrence_of(s2,turnon(Dispenser))
          & subactivity_occurrence(s1,o)
          & subactivity_occurrence(s2,o))
       ->
        (min_precedes(s1,s2,a))))).
```

Query 2d

In the following Otter syntax the verification query states:

For all occurrences of sanitize(Dispenser), and all subactivity occurrences $s_1$ and $s_2$ of activities $a_1$ and $a_2$, $s_1$ precedes $s_2$ in the sanitize(Dispenser) activity. This query was used in experiments 5.1.4, 5.1.16, 5.1.28, 5.2.4, 5.2.16, 5.2.28, 5.3.7, 5.3.22, and 5.3.37.

```
all o all s1 all s2 all a1 all a2
   (( occurrence_of(o,sanitize(Dispenser))
     & occurrence_of(s1,a1)
     & occurrence_of(s2,a2)
     & subactivity_occurrence(s1,o)
     & subactivity_occurrence(s2,o))
   ->
    (min_precedes(s1,s2,sanitize(Dispenser))))).
```
Query 3a

In the following Otter syntax the retrieval query states:

Does there exist an activity a, such that for any activity tree for a and all occurrences of
unplug(Dispenser) (s1), there does not exist an occurrence of turnon(Dispenser) (s2) that is
preceded by s1 in activity a? The query is to return the activity a that satisfies this statement.
This query was used in experiments 5.1.5, 5.1.17, 5.1.29, 5.2.5, 5.2.17, 5.2.29, 5.3.8, 5.3.23,
and 5.3.38.

exists a
    (activity(a)
      & (all s all s1
        (( root(s,a)
          & occurrence_of(s1,unplug(Dispenser))
          & min_precedes(s,s1,a))
        ->
          ( - (exists s2
               (occurrence_of(s2,turnon(Dispenser))
                & min_precedes(s1,s2,a))))))).

Query 3b

In the following Otter syntax the retrieval query states:

For any activity tree for sanitize(Dispenser), and all occurrences of a1 (s1), there does not
exist an occurrence of a subactivity of a (s2) that is preceded by s1. The query is to return
the activities that satisfy this statement. This query was used in experiments 5.1.6, 5.1.18,
5.1.30, 5.2.6, 5.2.18, 5.2.30, 5.3.9, 5.3.24, and 5.3.39.

all s all s1 all a1
    (( root(s,sanitize(Dispenser))
      & occurrence_of(s1,a1)
      & min_precedes(s,s1,sanitize(Dispenser)))
  ->
    ( - (exists s2 exists a2
         (subactivity(a2,a)
         & occurrence_of(s2,a1)
         & min_precedes(s1,s2,a2))))).

26
exists a 
(activity(a) 
& (all o all s1 
(( occurrence_of(o,a) 
 & occurrence_of(s1,unplug(Dispenser)) 
 & subactivity_occurrence(s1,o)) 
-> 
 ( - (exists s2 
 (occurrence_of(s2,turnon(Dispenser)) 
 & min_precedes(s1,s2,a)))))))).

Query 3d

In the following Otter syntax the retrieval query states:

For all occurrences of sanitize(Dispenser) and all subactivity occurrences (s1), there does not exist a subactivity occurrence (s2) that is preceded by s1 in sanitize(Dispenser). The query is to return the activities (a1, a2) that satisfy this statement. This query was used in experiments 5.1.8, 5.1.20, 5.1.32, 5.2.8, 5.2.20, 5.2.32, 5.3.11, 5.3.26, and 5.3.41.

all o all s1 all a1 
(( occurrence_of(o,sanitize(Dispenser)) 
 & occurrence_of(s1,a1) 
 & occurrence_of(s2,unplug(Dispenser)) 
 & min_precedes(s1,s2,a))))).
Query 4a

In the following Otter syntax the retrieval query states:

Does there exist an activity a, such that for any activity tree for a there exists an occurrence of unplug(Dispenser) (s1) that precedes an occurrence of turnon(Dispenser) (s2) in a? The query is to return the activity a that satisfies this statement. This query was used in experiments 5.1.9, 5.1.21, 5.1.33, 5.2.9, 5.2.21, 5.2.33, 5.3.12, 5.3.27, and 5.3.42.
Query 4b

In the following Otter syntax the retrieval query for subactivities states:

For any activity tree for sanitize(Dispenser), does there exist an occurrence of a1 (s1) and a subactivity a2, such that s1 precedes an occurrence of a2? The query is to return the values of a1 and a2 that satisfy this statement. This query was used in experiments 5.1.10, 5.1.22, 5.1.34, 5.2.10, 5.2.22, 5.2.34, 5.3.13, 5.3.28, and 5.3.43.

\[
\begin{align*}
&\text{all } s \\
&\quad \rightarrow \left( (\exists s1 \exists a1 \left( \text{occurrence}_{of}(s1, a1) \land \text{min}\text{\_precedes}(s, s1, \text{sanitize}(\text{Dispenser}))) \land (\exists s2 \exists a2 \left( \text{subactivity}(a2, a) \land \text{occurrence}_{of}(s2, a2) \land \text{min}\text{\_precedes}(s1, s2, \text{sanitize}(\text{Dispenser}))) \right)) \right).
\end{align*}
\]

Query 4c

In the following Otter syntax the retrieval query states:

Does there exist an activity a, such that for all occurrences of a there exists an occurrence of unplug(Dispenser) (s1) (which is a subactivity occurrence of o), which precedes an occurrence of turnon(Dispenser) (s2) in activity a? The query is to return the activity a that satisfies this statement. This query was used in experiments 5.1.11, 5.1.23, 5.1.35, 5.2.11, 5.2.23, 5.2.35, 5.3.14, 5.3.29, and 5.3.44.

\[
\begin{align*}
&\exists a \\
&\quad \left( \text{activity}(a) \land (\forall o \\
&\quad \left( \text{occurrence}_{of}(o, a) \land \left( (\exists s1 \left( \text{occurrence}_{of}(s1, \text{unplug}(\text{Dispenser})) \right) \right) \right) \right).
\end{align*}
\]
& subactivity_occurrence(s1,o))
& (exists s2
   (occurrence_of(s2,turnon(Dispenser))
   & min_precedes(s1,s2,a))))).

--------------------------------------------------------------------------------------------------------------

Query 4d

In the following Otter syntax the retrieval query for subactivities states:

For all occurrence of sanitize(Dispenser) (o), does there exist an occurrence of a1 (s1) (that is
a subactivity occurrence of o) and a subactivity a2, such that s1 precedes an occurrence of
a2? The query is to return the values of a1 and a2 that satisfy this statement. This query was
used in experiments 5.1.12, 5.1.24, 5.1.36, 5.2.12, 5.2.24, 5.2.36, 5.3.15, 5.3.30, and 5.3.45.

--------------------------------------------------------------------------------------------------------------

all o
  (occurrence_of(o,sanitize(Dispenser)))
->
  ((exists s1 exists a1
   (occurrence_of(s1,a1)
   & subactivity_occurrence(s1,o)))
  & (exists s2 exists a2
   (subactivity(a2,a)
   & occurrence_of(s2,a2)
   & min_precedes(s1,s2,sanitize(Dispenser)))))}.
4.4.2 Temporal Projections Queries

This section of the report describes additional queries that were developed but not used in any experimentation. They are provided as a guide for future experimentation on this project. There are four queries surrounding temporal projections, and three forms of each. The first forms are verification queries, the second are descriptive retrieval queries (e.g. what are the effects of this activity?), and the third are retrieval queries. These queries make use of two additional definitions, which are described below:

\[
\text{achieves}(s, f) \equiv (\neg \text{prior}(f, s) \land \text{holds}(f, s))
\]

\[
\text{falsifies}(s, f) \equiv (\text{prior}(f, s) \land \neg \text{holds}(f, s))
\]

The first one listed above states “An activity-occurrence achieves a fluent if the fluent does not hold before the activity-occurrence but holds after the activity-occurrence”. The second definition described above states “An activity-occurrence falsifies a fluent if the fluent holds before the occurrence but does not hold after the activity-occurrence”. The following are the temporal projection queries created:

**Query 5a**

In the following Otter syntax the verification query states:

After any occurrence of a, the state f holds.

```
all s all a all f
(leaf(s,a)
->
\text{holds}(f,s)).
```

---

31
Query 5b

In the following Otter syntax the descriptive query states:

For any occurrence of a, what state, f, holds? The query is to return the states (f) that satisfy this statement.

\[
\text{all } s \text{ all } a
\Rightarrow
(\text{exists } f \quad (\text{holds}(f, s)))
\]

Query 5c

In the following Otter syntax the retrieval query states:

Does there exist an activity a, that achieves the state f? The query is to return the activities (a) that satisfy this statement.

\[
\text{all } s \text{ all } f
\Rightarrow
(\text{exists } a \quad (\text{leaf}(s, a) \Rightarrow \text{holds}(f, s)))
\]

Query 6a

In the following Otter syntax the verification query states:

For any activity tree for a, there exists an occurrence that achieves the state f.

\[
\text{all } s1 \text{ all } a \text{ all } f
\Rightarrow
(\text{exists } s2 \quad (\text{leaf}(s2, a) \& \text{min_precedes}(s1, s2, a) \& \text{holds}(f, s2)))
\]
**Query 6b**

In the following Otter syntax the descriptive query states:

For any activity tree for a, there exists an occurrence that achieves what state, f? The query is to return the states (f) that satisfy this statement.

```
all s1 all a
  (root(s1,a)
   ->
    (exists s2 exists f
     (leaf(s2,a)
      & min_precedes(s1,s2,a)
      & holds(f,s2))))).
```

**Query 6c**

In the following Otter syntax the retrieval query states:

Does there exists an activity a, such that for any activity tree for a, there exist an occurrence that achieves the state f? The query is to return the activities (a) that satisfy this statement.

```
all s1 all f
  (exists a
    (root(s1,a)
     ->
      (exists s2
       (leaf(s2,a)
        & min_precedes(s1,s2,a)
        & holds(f,s2))))).
```
Query 7a

In the following Otter syntax the verification query states:

For any activity tree for a that achieves state f, there does not exist another occurrence (s2) that falsifies the state f.

```
all s all s1 all a all f
 (root(s,a)
  & achieves(s1,f)
  & min_precedes(s,s1,a)
 ->
  ( - (exists s2
       (falsifies(s2,f)
        & min_precedes(s1,s2,a))))).
```

-----------------------------

Query 7b

In the following Otter syntax the descriptive query states:

What state f has been achieved, for any activity tree for a, where there does not exist another occurrence (s2) that falsifies the state f? The query is to return the state f that satisfies this statement.

```
all s all s1 all a
 (exists f
  (root(s,a)
   & achieves(s1,f)
   & min_precedes(s,s1,a)
 ->
  ( - (exists s2
       (falsifies(s2,f)
        & min_precedes(s1,s2,a))))).
```

-----------------------------
Query 7c

In the following Otter syntax the retrieval query states:

Does there exist an activity tree for a that has achieved state f, where there does not exist another occurrence (s2) that falsifies the state f? The query is to return the value of a that satisfies this statement.

```
all s all s1 all a all f
(exists a
  (root(s,a)
   & achieves(s1,f)
   & min_precedes(s,s1,a)
  )
->
  ( - (exists s2
       (falsifies(s2,f)
        & min_precedes(s1,s2,a)))
  )).
```

Query 8a

In the following Otter syntax the verification query states:

For any activity tree for a, there exists an occurrence that achieves the state f, and there does not exist an occurrence which falsifies the state f.

```
all s all a all f
  (root(s,a)
  ->
  (exists s1
   (achieves(s1,f)
    & min_precedes(s,s1,a)
    & ( - (exists s2
         (falsifies(s2,f)
          & min_precedes(s1,s2,a))))))
  ).
```
Query 8b

In the following Otter syntax the descriptive query states:

For any activity tree for a, what state f is achieved by an occurrence, and there does not exist an occurrence which falsifies the state f? The query is to return the state f that satisfies this statement.

all s all a
   (root(s,a)
   ->
   (exists s1 exists f
    (achieves(s1,f)
     & min_precedes(s,s1,a)
     & ( - (exists s2
          (falsifies(s2,f)
           & min_precedes(s1,s2,a)))))))

Query 8c

In the following Otter syntax the retrieval query states:

Does there exist an activity tree for a that has achieved the state f, and there does not exist an occurrence which falsifies the state f? The query is to return the value of a that satisfies this statement.

all s all f
   (exists a
    (root(s,a)
   ->
   (exists s1
    (achieves(s1,f)
     & min_precedes(s,s1,a)
     & ( - (exists s2
          (falsifies(s2,f)
           & min_precedes(s1,s2,a))))))))
4.4.3 Preconditions and Effects Queries

This section of the report also describes additional queries that were developed but not used in any experimentation. They are provided as a guide for future experimentation on this project. There are three queries surrounding preconditions and effects, and similar to the temporal projection queries, there are three forms of each. The first forms are verification queries, the second are descriptive retrieval queries (e.g. what are the effects of this activity?), and the third are retrieval queries. The following are the precondition and effect queries created:

**Query 9a**

In the following Otter syntax the verification query states:
The state f must hold prior to all legal occurrences of a.

\[
\text{all } s \text{ all } a \text{ all } f \\
\quad \text{ (occurrence_of}(s,a) & \text{ legal}(s) \\
\quad \rightarrow \\
\quad \quad \text{ prior}(f,s)).
\]

**Query 9b**

In the following Otter syntax the descriptive query states:
What state, f, must hold prior to all legal occurrences of a? The query is to return the state (f) that satisfies this statement.

\[
\text{all } s \text{ all } a \\
\quad \text{ (occurrence_of}(s,a) & \text{ legal}(s) \\
\quad \rightarrow \\
\quad \quad (\text{exists } f \\
\quad \quad \quad \text{ (prior}(f,s)))).
\]
Query 9c

In the following Otter syntax the retrieval query states:

Does there exist an occurrence of a, prior to which the state f holds? The query is to return the activities (a) that satisfy this statement.

all s all f
(exists a
  (occurrence_of(s,a)
   & legal(s)
  ->
   prior(f,s))).

Query 10a

In the following Otter syntax the verification query states:

It is possible for a2 to occur after any legal occurrence of a1.

all s all a1 all a2
(occurrence_of(s,a1)
 & legal(s)
 ->
 poss(a2,s)).

Query 10b

In the following Otter syntax the descriptive query states:
What are the possible activities that can occur after a legal occurrence of a1? The query is to return the activities (a2) that satisfy this statement.

all s all a1
(occurrence_of(s,a1)
 & legal(s)
 ->
 (exists a2
  (poss(a2,s)))).
**Query 10c**
In the following Otter syntax the retrieval query states:
Does there exist a legal occurrence of a1 after which it is possible for a2 to occur? The query is to return the values of a1 that satisfy this statement.

```
all s all a2
(exists a1
  (occurrence_of(s,a1)
   & legal(s)
  ->
   poss(a2,s))).
```

**Query 11a**
In the following Otter syntax the verification query states:
For any activity tree for a, it is possible for a3 to occur after an occurrence of a2.

```
all s1 all s2 all a1 all a2 all a3
(root(s1,a1)
 & min_precedes(s1,s2,a)
 & occurrence_of(s2,a2)
 ->
 poss(a3,s2)).
```

**Query 11b**
In the following Otter syntax the descriptive query states:
For any activity tree for a1, what possible activity can occur after an occurrence of a2? The query is to return the value of a3 that satisfies this statement.

```
all s1 all s2 all a1 all a2 all a3
(root(s1,a1)
 & min_precedes(s1,s2,a)
 & occurrence_of(s2,a2)
 ->
 (exists a3
  (poss(a3,s2)))).
```
Query 11c

In the following Otter syntax the retrieval query states:

For any tree for \( a_1 \), does there exist an occurrence of \( a_2 \) after which it is possible for \( a_3 \) to occur? The query is to return the value \( a_2 \) that satisfies this statement.

\[
\text{all } s_1 \text{ all } s_2 \text{ all } a_1 \text{ all } a_3 \\
(\exists a_2 \\
\text{root}(s_1,a_1) \\
\& \text{min_precedes}(s_1,s_2,a) \\
\& \text{occurrence_of}(s_2,a_2) \\
\rightarrow \\
\text{poss}(a_3,s_2)).
\]
5.0 EXPERIMENTAL RESULTS

The following section outlines the experiments that were conducted for this project. Each summary outlines the theories incorporated in the reasoning process. As well, the queries that were used in each of the experiments and the results that Otter returned are indicated. Any proofs that were generated can be found in Appendix B.

Experiments 5.1

The following table provides an immediate outline identifying what background theories and form of process description each query is associated with and the corresponding experiment number. Below are the experiment summaries for Experiments 5.1.

Table 5.1. Outline of Experiments 5.1

<table>
<thead>
<tr>
<th>Query</th>
<th>$T_{actocc} \cup T_{complex} \cup \sum_{pd,complex} \cup T_{lemma_1}$</th>
<th>$T_{complex} \cup \sum_{pd,complex} \cup T_{lemma_1}$</th>
<th>$\sum_{pd,complex} \cup T_{lemma_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1b</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1c</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2a</td>
<td>5.1.1</td>
<td>5.1.13</td>
<td>5.1.25</td>
</tr>
<tr>
<td>2b</td>
<td>5.1.2</td>
<td>5.1.14</td>
<td>5.1.26</td>
</tr>
<tr>
<td>2c</td>
<td>5.1.3</td>
<td>5.1.15</td>
<td>5.1.27</td>
</tr>
<tr>
<td>2d</td>
<td>5.1.4</td>
<td>5.1.16</td>
<td>5.1.28</td>
</tr>
<tr>
<td>3a</td>
<td>5.1.5</td>
<td>5.1.17</td>
<td>5.1.29</td>
</tr>
<tr>
<td>3b</td>
<td>5.1.6</td>
<td>5.1.18</td>
<td>5.1.30</td>
</tr>
<tr>
<td>3c</td>
<td>5.1.7</td>
<td>5.1.19</td>
<td>5.1.31</td>
</tr>
<tr>
<td>3d</td>
<td>5.1.8</td>
<td>5.1.20</td>
<td>5.1.32</td>
</tr>
<tr>
<td>4a</td>
<td>5.1.9</td>
<td>5.1.21</td>
<td>5.1.33</td>
</tr>
<tr>
<td>4b</td>
<td>5.1.10</td>
<td>5.1.22</td>
<td>5.1.34</td>
</tr>
<tr>
<td>4c</td>
<td>5.1.11</td>
<td>5.1.23</td>
<td>5.1.35</td>
</tr>
<tr>
<td>4d</td>
<td>5.1.12</td>
<td>5.1.24</td>
<td>5.1.36</td>
</tr>
</tbody>
</table>
Experiment 5.1.1

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd,\text{complex}} \cup T_{\text{lemma}_1} \)

Query: \(-\exists a \\
(\text{activity}(a) \\
& (\forall s \forall s_1 \forall s_2 \\
 (\text{root}(s,a) \\
 & \text{occurrence_of}(s_1, \text{unplug(Dispenser)}) \\
 & \text{occurrence_of}(s_2, \text{turnon(Dispenser)}) \\
 & \text{min_precedes}(s,s_1,a) \\
 & \text{min_precedes}(s,s_2,a)) \\
\rightarrow \\
(\text{min_precedes}(s_1,s_2,a))))).\)

Results: Search stopped by max seconds option

Experiment 5.1.2

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd,\text{complex}} \cup T_{\text{lemma}_1} \)

Query: \(-\forall s \forall s_1 \forall s_2 \forall a_1 \forall a_2 \\
(\text{root}(s,s_1,\text{sanitize(Dispenser)}) \\
 & \text{occurrence_of}(s_1,a_1) \\
 & \text{occurrence_of}(s_2,a_2) \\
 & \text{min_precedes}(s,s_1,\text{sanitize(Dispenser)}) \\
 & \text{min_precedes}(s,s_2,\text{sanitize(Dispenser)}) \\
\rightarrow \\
(\text{min_precedes}(s_1,s_2,\text{sanitize(Dispenser)}))).\)

Results: Search stopped by max seconds option
Experiment 5.1.3

Theory: $T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_complex}} \cup T_{\text{lemma\_1}}$

Query: $\neg (\exists a$
\hspace{1em}(activity(a)
\hspace{1em}& (\forall o \forall s1 \forall s2
\hspace{1em}((\text{occurrence\_of}(o,a)
\hspace{1em}& \text{occurrence\_of}(s1,\text{unplug(Dispenser)})
\hspace{1em}& \text{occurrence\_of}(s2,\text{turnon(Dispenser)})
\hspace{1em}& \text{subactivity\_occurrence}(s1,o)
\hspace{1em}& \text{subactivity\_occurrence}(s2,o)
\hspace{1em})
\hspace{1em}->
\hspace{1em}(\text{min\_precedes}(s1,s2,a)))))).$

Results: Search stopped by max seconds option

Experiment 5.1.4

Theory: $T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_complex}} \cup T_{\text{lemma\_1}}$

Query: $\neg (\forall o \forall s1 \forall s2 \forall a1 \forall a2$
\hspace{1em}((\text{occurrence\_of}(o,\text{sanitize(Dispenser)})
\hspace{1em}& \text{occurrence\_of}(s1,a1)
\hspace{1em}& \text{occurrence\_of}(s2,a2)
\hspace{1em}& \text{subactivity\_occurrence}(s1,o)
\hspace{1em}& \text{subactivity\_occurrence}(s2,o)
\hspace{1em})
\hspace{1em}->
\hspace{1em}(\text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)})))).$

Results: Search stopped by max seconds option
Experiment 5.1.5

Theory: \( T_{\text{actocc} \cup \text{complex} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma_1}}} \)

Query: \(-\exists a \\
(\text{activity}(a) \\
& \forall s \forall s1 \\
(\text{root}(s,a) \\
& \text{occurrence_of}(s1,\text{unplug}(\text{Dispenser})) \\
& \text{min_precedes}(s,s1,a)) \\
\rightarrow \\
( - \exists s2 \\
(\text{occurrence_of}(s2,\text{turnon}(\text{Dispenser})) \\
& \text{min_precedes}(s1,s2,a)))))).\)

Results: Search stopped by max seconds option

Experiment 5.1.6

Theory: \( T_{\text{actocc} \cup \text{complex} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma_1}}} \)

Query: \(-\forall s \forall s1 \forall a1 \\
(\text{root}(s,\text{sanitize}(\text{Dispenser})) \\
& \text{occurrence_of}(s1,a1) \\
& \text{min_precedes}(s,s1,\text{sanitize}(\text{Dispenser}))) \\
\rightarrow \\
( - \exists s2 \exists a2 \\
(\text{subactivity}(a2,a) \\
& \text{occurrence_of}(s2,a2) \\
& \text{min_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})))))).\)

Results: Search stopped by max seconds option
Experiment 5.1.7

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_complex} \cup T_{\text{lemma}_1} \)

Query: \neg(\exists a \\
(\text{activity}(a) \\
& (\forall o \forall s1 \\
((\text{occurrence}_o(o,a) \\
& \text{occurrence}_o(s1,\text{unplug}(\text{Dispenser})) \\
& \text{subactivity}_o(s1,o)) \\
\rightarrow \\
(\neg(\exists s2 \\
(\text{occurrence}_o(s2,\text{turnon}(\text{Dispenser})) \\
& \text{min\_precedes}(s1,s2,a)))))))

Results: Search stopped by max seconds option

Experiment 5.1.8

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_complex} \cup T_{\text{lemma}_1} \)

Query: \neg(\forall o \forall s1 \forall a1 \\
(\neg(\text{occurrence}_o(o,\text{sanitize}(\text{Dispenser})) \\
& \text{occurrence}_o(s1,a1) \\
& \text{subactivity}_o(s1,o)) \\
\rightarrow \\
(\neg(\exists s2 \exists a2 \\
(\text{subactivity}(a2,a) \\
& \text{occurrence}_o(s2,a2) \\
& \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})))")

Results: Search stopped by max seconds option
Experiment 5.1.9

Theory: \( T_{actocc \cup complex} \cup \sum_{pd_{complex}} \cup T_{lemma_1} \)

Query: \(- (\exists a \\ (activity(a) \& (\forall s \ (root(s,a) => ((\exists s_1 (\text{occurrence} \_of(s_1,\text{unplug(Dispenser)}) \& \text{min}_\text{precedes}(s,s_1,a)) \& (\exists s_2 (\text{occurrence} \_of(s_2,\text{turnon(Dispenser)}) \& \text{min}_\text{precedes}(s_1,s_2,a))))))))).\)

Results: Search stopped by max seconds option

Experiment 5.1.10

Theory: \( T_{actocc \cup complex} \cup \sum_{pd_{complex}} \cup T_{lemma_1} \)

Query: \(- (\forall s \ (root(s,\text{sanitize(Dispenser)}) => ((\exists s_1 \exists a_1 (\text{occurrence} \_of(s_1,a_1) \& \text{min}_\text{precedes}(s,s_1,\text{sanitize(Dispenser)}))) \& (\exists s_2 \exists a_2 (\text{subactivity}(a_2,a) \& \text{occurrence} \_of(s_2,a_2) \& \text{min}_\text{precedes}(s_1,s_2,\text{sanitize(Dispenser)})))))).\)

Results: Search stopped by max seconds option
Experiment 5.1.11

**Theory:** \( T_{actocc} \cup complex \cup \sum_{pd\_complex} \cup T_{lemma\_1} \)

**Query:**
\[
- \left( \exists a \\
\quad \left( \text{activity}(a) \& \left( \forall o \\
\quad \left( \text{occurrence\_of}(o,a) \rightarrow \\
\quad \quad \left( \exists s_1 \\
\quad \quad \quad \left( \text{occurrence\_of}(s_1,\text{unplug(Dispenser)}) \& \text{subactivity\_occurrence}(s_1,o) \right) \& \left( \exists s_2 \\
\quad \quad \quad \quad \left( \text{occurrence\_of}(s_2,\text{turnon(Dispenser)}) \& \text{min\_precedes}(s_1,s_2,a) \right) \right) \right) \right) \right) \right).
\]

**Results:** Search stopped by max seconds option

Experiment 5.1.12

**Theory:** \( T_{actocc} \cup complex \cup \sum_{pd\_complex} \cup T_{lemma\_1} \)

**Query:**
\[
- \left( \forall o \\
\quad \left( \text{occurrence\_of}(o,\text{sanitize(Dispenser)}) \rightarrow \\
\quad \quad \left( \exists s_1 \exists a_1 \\
\quad \quad \quad \left( \text{occurrence\_of}(s_1,a_1) \& \text{subactivity\_occurrence}(s_1,o) \right) \& \left( \exists s_2 \exists a_2 \\
\quad \quad \quad \quad \left( \text{subactivity}(a_2,a) \& \text{occurrence\_of}(s_2,a_2) \& \text{min\_precedes}(s_1,s_2,\text{sanitize(Dispenser)}) \right) \right) \right) \right) \right).
\]

**Results:** Search stopped by max seconds option
Experiment 5.1.13

**Theory:** \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}_1} \)

**Query:** 
\[-(\text{exists } a \\
\quad (\text{activity}(a) \\
\quad \& (\text{all } s \text{ all } s1 \text{ all } s2 \\
\quad \quad ((\text{root}(s,a) \\
\quad \quad \& \text{occurrence_of}(s1,\text{unplug(Dispenser)}) \\
\quad \quad \& \text{occurrence_of}(s2,\text{turnon(Dispenser)}) \\
\quad \quad \& \text{min_precedes}(s,s1,a) \\
\quad \quad \& \text{min_precedes}(s,s2,a)) \\
\quad \quad -> \\
\quad \quad (\text{min_precedes}(s1,s2,a)))))).\]

**Results:** Search stopped because SOS empty

Experiment 5.1.14

**Theory:** \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}_1} \)

**Query:** 
\[-(\text{all } s \text{ all } s1 \text{ all } s2 \text{ all } a1 \text{ all } a2 \\
\quad ((\text{root}(s,\text{sanitize(Dispenser)}) \\
\quad \& \text{occurrence_of}(s1,a1) \\
\quad \& \text{occurrence_of}(s2,a2) \\
\quad \& \text{min_precedes}(s,s1,\text{sanitize(Dispenser)}) \\
\quad \& \text{min_precedes}(s,s2,\text{sanitize(Dispenser)}) \\
\quad ) \\
\quad -> \\
\quad (\text{min_precedes}(s1,s2,\text{sanitize(Dispenser)})])).\]

**Results:** Search stopped because SOS empty
Experiment 5.1.15

Theory: \[ T_{\text{complex}} \cup \sum_{\text{pd\_complex}} \cup T_{\text{lemma\_1}} \]

Query: \[-(\exists a (\text{activity}(a) \& (\forall o \forall s1 \forall s2 ((\text{occurrence\_of}(o, a) \& \text{occurrence\_of}(s1, \text{unplug}(\text{Dispenser})) \& \text{occurrence\_of}(s2, \text{turnon}(\text{Dispenser})) \& \text{subactivity\_occurrence}(s1, o) \& \text{subactivity\_occurrence}(s2, o) ) -> (\text{min\_precedes}(s1, s2, a)))))).\]

Results: Search stopped because SOS empty

Experiment 5.1.16

Theory: \[ T_{\text{complex}} \cup \sum_{\text{pd\_complex}} \cup T_{\text{lemma\_1}} \]

Query: \[-(\forall o \forall s1 \forall s2 \forall a1 \forall a2 ((\text{occurrence\_of}(o, \text{sanitize}(\text{Dispenser})) \& \text{occurrence\_of}(s1, a1) \& \text{occurrence\_of}(s2, a2) \& \text{subactivity\_occurrence}(s1, o) \& \text{subactivity\_occurrence}(s2, o) ) -> (\text{min\_precedes}(s1, s2, \text{sanitize}(\text{Dispenser}))))).\]

Results: Search stopped because SOS empty
Experiment 5.1.17

Theory: \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}} \)

Query: 
\[
\neg (\exists a \\
\quad (\text{activity}(a) \\
\quad \& (\forall s \forall s_1 \\
\quad \quad (\text{root}(s,a) \\
\quad \quad \& \text{occurrence_of}(s_1,\text{unplug}(\text{Dispenser})) \\
\quad \quad \& \text{min_precedes}(s,s_1,a)) \\
\quad \Rightarrow \\
\quad \quad (\neg (\exists s_2 \\
\quad \quad \quad (\text{occurrence_of}(s_2,\text{turnon}(\text{Dispenser})) \\
\quad \quad \quad \& \text{min_precedes}(s_1,s_2,a)))))))).
\]

Results: Search stopped because SOS empty

Experiment 5.1.18

Theory: \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}} \)

Query: 
\[
\neg (\forall s \forall s_1 \forall a_1 \\
\quad ((\text{root}(s,\text{sanitize}(\text{Dispenser})) \\
\quad \& \text{occurrence_of}(s_1,a_1) \\
\quad \& \text{min_precedes}(s,s_1,\text{sanitize}(\text{Dispenser}))) \\
\quad \Rightarrow \\
\quad \quad (\neg (\exists s_2 \exists a_2 \\
\quad \quad \quad (\text{subactivity}(a_2,a) \\
\quad \quad \quad \& \text{occurrence_of}(s_2,a_2) \\
\quad \quad \quad \& \text{min_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser}))) \\
\quad \quad ))))).
\]

Results: Search stopped because SOS empty
Experiment 5.1.19

Theory: \[ T_{complex} \cup \sum_{pd_{complex}} \cup T_{lemma_1} \]

Query: \[-(\exists a \quad (activity(a) \& (\forall o \forall s1 \quad ((occurrence_of(o,a) \& occurrence_of(s1,unplug(Dispenser)) \& subactivity_occurrence(s1,o)) \Rightarrow (\neg (\exists s2 \quad (occurrence_of(s2,turnon(Dispenser)) \& min_precedes(s1,s2,a)))))),\).

Results: Search stopped because SOS empty

Experiment 5.1.20

Theory: \[ T_{complex} \cup \sum_{pd_{complex}} \cup T_{lemma_1} \]

Query: \[-(\forall o \forall s1 \forall a1 \quad ((occurrence_of(o,sanitize(Dispenser)) \& occurrence_of(s1,a1) \& subactivity_occurrence(s1,o)) \Rightarrow (\neg (\exists s2 \exists a2 \quad (subactivity(a2,a) \& occurrence_of(s2,a2) \& min_precedes(s1,s2,sanitize(Dispenser))))))),\).

Results: Search stopped because SOS empty
Experiment 5.1.21

Theory: \( T_{\text{complex}} \cup \sum_{pd\_complex} \cup T_{\text{lemma}1} \)

Query: \(- (\exists a (\text{activity}(a) \& (\forall s (\text{root}(s,a) \rightarrow ((\exists s_1 (\text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser})) \& \text{min\_precedes}(s,s_1,a)) \& (\exists s_2 (\text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser})) \& \text{min\_precedes}(s_1,s_2,a)))))\)).\)

Results: Search stopped because SOS empty

Experiment 5.1.22

Theory: \( T_{\text{complex}} \cup \sum_{pd\_complex} \cup T_{\text{lemma}1} \)

Query: \(- (\forall s (\text{root}(s,\text{sanitize}(\text{Dispenser})) \rightarrow ((\exists s_1 (\exists a_1 (\text{occurrence\_of}(s_1,a_1) \& \text{min\_precedes}(s,s_1,\text{sanitize}(\text{Dispenser})))) \& (\exists s_2 (\exists a_2 (\text{subactivity}(a_2,a) \& \text{occurrence\_of}(s_2,a_2) \& \text{min\_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser}))))))).\)

Results: Search stopped because SOS empty
Experiment 5.1.23

 Theory: \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}_1} \)

 Query: \(-\exists a (\text{activity}(a) \land \forall o (\text{occurrence_of}(o,a) \rightarrow ((\exists s_1 (\text{occurrence_of}(s_1,\text{unplug(Dispenser)}) \land \text{subactivity_occurrence}(s_1,o)) \land (\exists s_2 (\text{occurrence_of}(s_2,\text{turnon(Dispenser)}) \land \text{min_precedes}(s_1,s_2,a)))))))\).

 Results: Search stopped because SOS empty

Experiment 5.1.24

 Theory: \( T_{\text{complex}} \cup \sum_{pd_{\text{complex}}} \cup T_{\text{lemma}_1} \)

 Query: \(-\forall o (\text{occurrence_of}(o,\text{sanitize(Dispenser)}) \rightarrow ((\exists s_1 \exists a_1 (\text{occurrence_of}(s_1,a_1) \land \text{subactivity_occurrence}(s_1,o))) \land (\exists s_2 \exists a_2 (\text{subactivity}(a_2,a) \land \text{occurrence_of}(s_2,a_2) \land \text{min_precedes}(s_1,s_2,\text{sanitize(Dispenser)})))))\).

 Results: Search stopped because SOS empty
Experiment 5.1.25

Theory: \( \sum_{pd\_complex} \cup T_{lemma\_2} \)

Query: \(- (\exists a \\
(\text{activity}(a) \\
& (\forall s \forall s1 \forall s2 \\
( ( \text{root}(s,a) \\
& \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \\
& \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \\
& \text{min\_precedes}(s,s1,a) \\
& \text{min\_precedes}(s,s2,a)) \\
-> \\
(\text{min\_precedes}(s1,s2,a)))))). \)

Results: Search stopped because SOS empty

Experiment 5.1.26

Theory: \( \sum_{pd\_complex} \cup T_{lemma\_2} \)

Query: \(- (\forall s \forall s1 \forall s2 \forall a1 \forall a2 \\
( ( \text{root}(s,\text{sanitize}(\text{Dispenser})) \\
& \text{occurrence\_of}(s1,a1) \\
& \text{occurrence\_of}(s2,a2) \\
& \text{min\_precedes}(s,s1,\text{sanitize}(\text{Dispenser})) \\
& \text{min\_precedes}(s,s2,\text{sanitize}(\text{Dispenser})) \\
) \\
-> \\
(\text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})))). \)

Results: Search stopped because SOS empty
Experiment 5.1.27

Theory: $\sum_{pd\text{-complex}} \bigcup T_{\text{lemma}_2}$

Query: $\neg \exists a$

(activity(a)
& (all o all s1 all s2
(( occurrence_of(o,a)
& occurrence_of(s1,unplug(Dispenser))
& occurrence_of(s2,turnon(Dispenser))
& subactivity_occurrence(s1,o)
& subactivity_occurrence(s2,o)
)
->
(min_precedes(s1,s2,a)))))).

Results: Search stopped because SOS empty

Experiment 5.1.28

Theory: $\sum_{pd\text{-complex}} \bigcup T_{\text{lemma}_2}$

Query: $\neg (all o all s1 all s2 all a1 all a2$

(( occurrence_of(o,sanitize(Dispenser))
& occurrence_of(s1,a1)
& occurrence_of(s2,a2)
& subactivity_occurrence(s1,o)
& subactivity_occurrence(s2,o)
)
->
(min_precedes(s1,s2,sanitize(Dispenser))))).

Results: Search stopped because SOS empty
Experiment 5.1.29

Theory: \[ \sum_{pd\_complex} \cup T_{\text{lemma}2} \]

Query: \[-(\exists a \quad \\
\quad (\text{activity}(a) \quad \\
\quad \& \quad (\forall s \quad \forall s1 \quad \\
\quad (\text{root}(s,a) \quad \\
\quad \& \quad \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \quad \\
\quad \& \quad \text{min\_precedes}(s,s1,a)) \quad \\
\quad \rightarrow \quad \\
\quad (\neg (\exists s2 \quad \\
\quad (\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \quad \\
\quad \& \quad \text{min\_precedes}(s1,s2,a)))))).\]

Results: Search stopped because SOS empty

Experiment 5.1.30

Theory: \[ \sum_{pd\_complex} \cup T_{\text{lemma}2} \]

Query: \[-(\forall s \quad \forall s1 \quad \forall a1 \quad \\
\quad ((\text{root}(s,\text{sanitize}(\text{Dispenser})) \quad \\
\quad \& \quad \text{occurrence\_of}(s1,a1) \quad \\
\quad \& \quad \text{min\_precedes}(s,s1,\text{sanitize}(\text{Dispenser}))) \quad \\
\quad \rightarrow \quad \\
\quad (\neg (\exists s2 \exists a2 \quad \\
\quad (\text{subactivity}(a2,a) \quad \\
\quad \& \quad \text{occurrence\_of}(s2,a2) \quad \\
\quad \& \quad \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser}))) \quad \\
\quad ))))).\]

Results: Search stopped because SOS empty
Experiment 5.1.31

Theory: \[ \sum_{pd\_complex} \cup T_{\text{lemma}_2} \]

Query: \[ \neg (\exists a \\
(\text{activity}(a) \\
& (\forall o \forall s1 \\
(( \text{occurrence\_of}(o,a) \\
& \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \\
& \text{subactivity\_occurrence}(s1,o)) \\
\rightarrow \\
( - (\exists s2 \\
(\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \\
& \text{min\_precedes}(s1,s2,a)))))))))). \]

Results: Search stopped because SOS empty

Experiment 5.1.32

Theory: \[ \sum_{pd\_complex} \cup T_{\text{lemma}_2} \]

Query: \[ \neg (\forall o \forall s1 \forall a1 \\
(( \text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \\
& \text{occurrence\_of}(s1,a1) \\
& \text{subactivity\_occurrence}(s1,o)) \\
\rightarrow \\
( - (\exists s2 \exists a2 \\
(\text{subactivity}(a2,a) \\
& \text{occurrence\_of}(s2,a2) \\
& \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \\
)))))). \]

Results: Search stopped because SOS empty
Experiment 5.1.33

Theory: \( \sum_{pd\_complex} \cup T_{\text{lemma}\_2} \)

Query: \(-\exists a \\
\quad (\text{activity}(a) \\
\quad \& \ (\forall s \\
\quad \ (\text{root}(s,a) \\
\quad \rightarrow \\
\quad \ (\exists s_1 \\
\quad \ (\text{occurrence\_of}(s_1, \text{unplug}(\text{Dispenser})) \\
\quad \& \ \text{min\_precedes}(s,s_1,a)) \\
\quad \& \ (\exists s_2 \\
\quad \ (\text{occurrence\_of}(s_2, \text{turnon}(\text{Dispenser})) \\
\quad \& \ \text{min\_precedes}(s_1,s_2,a))))))\).

Results: Search stopped because SOS empty

Experiment 5.1.34

Theory: \( \sum_{pd\_complex} \cup T_{\text{lemma}\_2} \)

Query: \(-\forall s \\
\quad (\text{root}(s,\text{sanitize}(\text{Dispenser})) \\
\quad \rightarrow \\
\quad \ (\exists s_1 \exists a_1 \\
\quad \ (\text{occurrence\_of}(s_1,a_1) \\
\quad \& \ \text{min\_precedes}(s,s_1,\text{sanitize}(\text{Dispenser})))) \\
\quad \& \ (\exists s_2 \exists a_2 \\
\quad \ (\text{subactivity}(a_2,a) \\
\quad \& \ \text{occurrence\_of}(s_2,a_2) \\
\quad \& \ \text{min\_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser}))))))\).

Results: Search stopped because SOS empty
Experiment 5.1.35

Theory: \( \sum_{pd\_complex} \cup T_{\text{lemma}_2} \)

Query: \(-'(\exists a
\quad (\text{activity}(a)
\quad \& \quad (\forall o
\quad (\text{occurrence\_of}(o,a)
\quad \rightarrow
\quad ((\exists s_1
\quad (\text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser}))
\quad \& \quad \text{subactivity\_occurrence}(s_1,o))
\quad \& \quad (\exists s_2
\quad (\text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser}))
\quad \& \quad \text{min\_precedes}(s_1,s_2,a))))))))\).

Results: Search stopped because SOS empty

Experiment 5.1.36

Theory: \( \sum_{pd\_complex} \cup T_{\text{lemma}_2} \)

Query: \(-'(\forall o
\quad (\text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser}))
\quad \rightarrow
\quad ((\exists s_1 \exists a_1
\quad (\text{occurrence\_of}(s_1,a_1)
\quad \& \quad \text{subactivity\_occurrence}(s_1,o))
\quad \& \quad (\exists s_2 \exists a_2
\quad (\text{subactivity}(a_2,a)
\quad \& \quad \text{occurrence\_of}(s_2,a_2)
\quad \& \quad \text{min\_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser}))
\quad ))))).\)

Results: Search stopped because SOS empty
### Experiments 5.2

The following table provides an immediate outline identifying what background theories and form of process description each query is associated with and the corresponding experiment number. Below are the experiment summaries for Experiments 5.2.

#### Table 5.2. Outline of Experiments 5.2

<table>
<thead>
<tr>
<th>Query</th>
<th>$T_{actoc} \cup \text{complex} \cup \sum_{pd_closure} \cup T_{lemma_2}$</th>
<th>$T_{complex} \cup \sum_{pd_closure} \cup T_{lemma_2}$</th>
<th>$\sum_{pd_closure} \cup T_{lemma_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1b</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1c</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2a</td>
<td>5.2.1</td>
<td>5.2.13</td>
<td>5.2.25</td>
</tr>
<tr>
<td>2b</td>
<td>5.2.2</td>
<td>5.2.14</td>
<td>5.2.26</td>
</tr>
<tr>
<td>2c</td>
<td>5.2.3</td>
<td>5.2.15</td>
<td>5.2.27</td>
</tr>
<tr>
<td>2d</td>
<td>5.2.4</td>
<td>5.2.16</td>
<td>5.2.28</td>
</tr>
<tr>
<td>3a</td>
<td>5.2.5</td>
<td>5.2.17</td>
<td>5.2.29</td>
</tr>
<tr>
<td>3b</td>
<td>5.2.6</td>
<td>5.2.18</td>
<td>5.2.30</td>
</tr>
<tr>
<td>3c</td>
<td>5.2.7</td>
<td>5.2.19</td>
<td>5.2.31</td>
</tr>
<tr>
<td>3d</td>
<td>5.2.8</td>
<td>5.2.20</td>
<td>5.2.32</td>
</tr>
<tr>
<td>4a</td>
<td>5.2.9</td>
<td>5.2.21</td>
<td>5.2.33</td>
</tr>
<tr>
<td>4b</td>
<td>5.2.10</td>
<td>5.2.22</td>
<td>5.2.34</td>
</tr>
<tr>
<td>4c</td>
<td>5.2.11</td>
<td>5.2.23</td>
<td>5.2.35</td>
</tr>
<tr>
<td>4d</td>
<td>5.2.12</td>
<td>5.2.24</td>
<td>5.2.36</td>
</tr>
</tbody>
</table>
Experiment 5.2.1

**Theory:** \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_\text{closure}} \cup T_{\text{lemma}_2} \)

**Query:** \(-\exists a \\
\quad (\text{activity}(a) \\
\quad \& \ (\forall s \forall s_1 \forall s_2 \forall a_1 \forall a_2 \\
\quad \quad (( \text{root}(s,a) \\
\quad \quad \& \ \text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser})) \\
\quad \quad \& \ \text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser})) \\
\quad \quad \& \ \text{min\_precedes}(s,s_1,a) \\
\quad \quad \& \ \text{min\_precedes}(s,s_2,a)) \\
\quad \quad \rightarrow \\
\quad \quad (\text{min\_precedes}(s_1,s_2,a))))))\).

**Results:** Search stopped by max seconds option

Experiment 5.2.2

**Theory:** \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_\text{closure}} \cup T_{\text{lemma}_2} \)

**Query:** \(-\forall s \forall s_1 \forall s_2 \forall a_1 \forall a_2 \\
\quad (( \text{root}(s,\text{sanitize}(\text{Dispenser})) \\
\quad \& \ \text{occurrence\_of}(s_1,a_1) \\
\quad \& \ \text{occurrence\_of}(s_2,a_2) \\
\quad \& \ \text{min\_precedes}(s,s_1,\text{sanitize}(\text{Dispenser})) \\
\quad \& \ \text{min\_precedes}(s,s_2,\text{sanitize}(\text{Dispenser})) \\
\quad \rightarrow \\
\quad (\text{min\_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser})))).\)

**Results:** Search stopped by max seconds option
Experiment 5.2.3

Theory: \( T_{actocc} \cup \text{complex} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \[-\exists a \\
\text{(activity}(a) \\
\& \ (\forall o \forall s1 \forall s2 \\
(\text{occurrence\_of}(o,a) \\
\& \ \text{occurrence\_of}(s1,\text{unplug(Dispenser)})) \\
\& \ \text{occurrence\_of}(s2,\text{turnon(Dispenser)})) \\
\& \ \text{subactivity\_occurrence}(s1,o) \\
\& \ \text{subactivity\_occurrence}(s2,o) \\
) \\
\rightarrow \\
(\text{min\_precedes}(s1,s2,a)))) \].

Results: Search stopped by max seconds option

Experiment 5.2.4

Theory: \( T_{actocc} \cup \text{complex} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \[-(\forall o \forall s1 \forall s2 \forall a1 \forall a2 \\
(\text{occurrence\_of}(o,\text{sanitize(Dispenser)})) \\
\& \ \text{occurrence\_of}(s1,a1) \\
\& \ \text{occurrence\_of}(s2,a2) \\
\& \ \text{subactivity\_occurrence}(s1,o) \\
\& \ \text{subactivity\_occurrence}(s2,o) \\
) \\
\rightarrow \\
(\text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)})))) \].

Results: Search stopped by max seconds option
Experiment 5.2.5

Theory: \( T_{actocc} \cup complex \cup \sum_{pd\_closure} \cup T_{lemma\_2} \)

Query: \[-(\exists a \ (activity(a) \& (\forall s \forall s1 \ ((\ root(s,a) \& occurrence\_of(s1,unplug(Dispenser)) \& min\_precedes(s,s1,a)) -> (\neg(\exists s2 \ (occurrence\_of(s2,turnon(Dispenser)) \& min\_precedes(s1,s2,a)))))))).\]

Results: Search stopped by max seconds option

Experiment 5.2.6

Theory: \( T_{actocc} \cup complex \cup \sum_{pd\_closure} \cup T_{lemma\_2} \)

Query: \[-(\forall s \forall s1 \forall a1 \ ((\ root(s,sanitize(Dispenser)) \& occurrence\_of(s1,a1) \& min\_precedes(s,s1,sanitize(Dispenser))) -> (\neg(\exists s2 \exists a2 \ (subactivity(a2,a) \& occurrence\_of(s2,a2) \& min\_precedes(s1,s2,sanitize(Dispenser)))))))).\]

Results: Search stopped by max seconds option
Experiment 5.2.7

Theory:  \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_cloure} \cup T_{\text{lemma\_2}} \)

Query:  
\[-(\exists a \\
\quad (\text{activity}(a) \\
\quad \& \ (\forall o \forall s1 \\
\quad \quad (\text{occurrence\_of}(o,a) \\
\quad \quad \& \ \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \\
\quad \quad \& \ \text{subactivity\_occurrence}(s1,o)) \\
\quad \quad \rightarrow \\
\quad \quad \quad ( - (\exists s2 \\
\quad \quad \quad \quad (\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \\
\quad \quad \quad \quad \& \ \text{min\_precedes}(s1,s2,a)))))))].

Results:  Search stopped by max seconds option

Experiment 5.2.8

Theory:  \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_cloure} \cup T_{\text{lemma\_2}} \)

Query:  
\[-(\forall o \forall s1 \forall a1 \\
\quad ((\text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \\
\quad \& \ \text{occurrence\_of}(s1,a1) \\
\quad \& \ \text{subactivity\_occurrence}(s1,o)) \\
\quad \rightarrow \\
\quad \quad ( - (\exists s2 \exists a2 \\
\quad \quad \quad (\text{subactivity}(a2,a) \\
\quad \quad \quad \& \ \text{occurrence\_of}(s2,a2) \\
\quad \quad \quad \& \ \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \\
\quad \quad \quad ))))).

Results:  Search stopped by max seconds option
Experiment 5.2.9

Theory:  \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd \_\text{closure}} \cup T_{\text{lemma} \_2} \)

Query:  \(-\exists a
\quad (\text{activity}(a)
\quad \& \quad (\forall s
\quad (\text{root}(s,a)
\quad \rightarrow
\quad \exists s1
\quad (\text{occurrence\_of}(s1,\text{unplug(Dispenser)})
\quad \& \quad \text{min\_precedes}(s,s1,a))
\quad \& \quad \exists s2
\quad (\text{occurrence\_of}(s2,\text{turnon(Dispenser)})
\quad \& \quad \text{min\_precedes}(s1,s2,a)))))) \).

Results:  Search stopped by max seconds option

Experiment 5.2.10

Theory:  \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd \_\text{closure}} \cup T_{\text{lemma} \_2} \)

Query:  \(-\forall s
\quad (\text{root}(s,\text{sanitize(Dispenser)})
\quad \rightarrow
\quad \exists s1 \exists a1
\quad (\text{occurrence\_of}(s1,a1)
\quad \& \quad \text{min\_precedes}(s,s1,\text{sanitize(Dispenser)}))
\quad \& \quad \exists s2 \exists a2
\quad (\text{subactivity}(a2,a)
\quad \& \quad \text{occurrence\_of}(s2,a2)
\quad \& \quad \text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)}))))) \).

Results:  Search stopped by max seconds option
Experiment 5.2.11

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_closure}} \cup T_{\text{lemma\_2}} \)

Query: \( \neg (\exists a \\
(\text{activity}(a) \\
\& (\forall o \\
(\text{occurrence\_of}(o,a) \\
\rightarrow \\
((\exists s1 \\
(\text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \\
\& \text{subactivity\_occurrence}(s1,o)) \\
\& (\exists s2 \\
(\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \\
\& \text{min\_precedes}(s1,s2,a)))))))) \\
\).

Results: Search stopped by max seconds option

Experiment 5.2.12

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_closure}} \cup T_{\text{lemma\_2}} \)

Query: \( \neg (\forall o \\
(\text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \\
\rightarrow \\
((\exists s1 \exists a1 \\
(\text{occurrence\_of}(s1,a1) \\
\& \text{subactivity\_occurrence}(s1,o)) \\
\& (\exists s2 \exists a2 \\
(\text{subactivity}(a2,a) \\
\& \text{occurrence\_of}(s2,a2) \\
\& \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \\
)))))).

Results: Search stopped by max seconds option
Experiment 5.2.13

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}2} \)

Query: \(-\exists a \\\n& (\text{activity}(a) \\
& \& (\text{all } s \text{ all } s1 \text{ all } s2 \\
& ( ( \text{root}(s,a) \\
& \& \text{occurrence\_of}(s1, \text{unplug(Dispenser)}) \\
& \& \text{occurrence\_of}(s2, \text{turnon(Dispenser)}) \\
& \& \text{min\_precedes}(s,s1,a) \\
& \& \text{min\_precedes}(s,s2,a) \\
\rightarrow \\
\text{min\_precedes}(s1,s2,a)))))\).

Results: Search stopped by max seconds option

Experiment 5.2.14

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}2} \)

Query: \(-\forall s \text{ all } s1 \text{ all } s2 \text{ all } a1 \text{ all } a2 \\\n& ( ( \text{root}(s,\text{sanitize(Dispenser)}) \\
& \& \text{occurrence\_of}(s1,a1) \\
& \& \text{occurrence\_of}(s2,a2) \\
& \& \text{min\_precedes}(s,s1,\text{sanitize(Dispenser)}) \\
& \& \text{min\_precedes}(s,s2,\text{sanitize(Dispenser)}) \\
) \\
\rightarrow \\
\text{min\_precedes}(s1,s2,\text{sanitize(Dispenser))))\).

Results: Search stopped by max seconds option
Experiment 5.2.15

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \(-\left( \exists a \left( \text{activity}(a) \& (\forall o \forall s1 \forall s2 \left( \left( \text{occurrence\_of}(o,a) \& \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \& \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \& \text{subactivity\_occurrence}(s1,o) \& \text{subactivity\_occurrence}(s2,o) \right) \implies (\text{min\_precedes}(s1,s2,a)) \right) \right) \right)\).

Results: Search stopped by max seconds option

Experiment 5.2.16

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \(-\left( \forall o \forall s1 \forall s2 \forall a1 \forall a2 \left( \left( \text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \& \text{occurrence\_of}(s1,a1) \& \text{occurrence\_of}(s2,a2) \& \text{subactivity\_occurrence}(s1,o) \& \text{subactivity\_occurrence}(s2,o) \right) \implies (\text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser}))) \right) \right)\).

Results: Search stopped by max seconds option
Experiment 5.2.17

Theory: $T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2}$

Query: $-\exists a$

(activity(a)
& (\forall s \forall s1 ((\text{root}(s,a)
& \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser}))
& \text{min\_precedes}(s,s1,a))$
->
( - (\exists s2
(\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser}))
& \text{min\_precedes}(s1,s2,a)))))))).

Results: Search stopped by max seconds option

Experiment 5.2.18

Theory: $T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2}$

Query: $-\forall s \forall s1 \forall a1$

((\text{root}(s,\text{sanitize}(\text{Dispenser}))
& \text{occurrence\_of}(s1,a1)
& \text{min\_precedes}(s,s1,\text{sanitize}(\text{Dispenser})))$
->
( - (\exists s2 \exists a2
(subactivity(a2,a)
& \text{occurrence\_of}(s2,a2)
& \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})))))))).

Results: Search stopped by max seconds option
Experiment 5.2.19

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \( \neg (\text{exists } a \\
(\text{activity}(a) \\
\& \forall o \forall s1 \\
(\text{occurrence\_of}(o,a) \\
\& \text{occurrence\_of}(s1,\text{unplug(Dispenser)}) \\
\& \text{subactivity\_occurrence}(s1,o)) \\
\rightarrow \\
(\neg (\text{exists } s2 \\
(\text{occurrence\_of}(s2,\text{turnon(Dispenser)}) \\
\& \text{min\_precedes}(s1,s2,a)))))))) \).

Results: Search stopped by max seconds option

Experiment 5.2.20

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

Query: \( \neg (\forall o \forall s1 \forall a1 \\
(\text{occurrence\_of}(o,\text{sanitize(Dispenser)}) \\
\& \text{occurrence\_of}(s1,a1) \\
\& \text{subactivity\_occurrence}(s1,o)) \\
\rightarrow \\
(\neg (\text{exists } s2 \exists a2 \\
(\text{subactivity}(a2,a) \\
\& \text{occurrence\_of}(s2,a2) \\
\& \text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)}))))) \).

Results: Search stopped by max seconds option
Experiment 5.2.21

**Theory:**  
\[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \]

**Query:**  
\[-(\exists a \ (\text{activity}(a) \land \forall s \ (\text{root}(s,a) \rightarrow ((\exists s_1 \ (\text{occurrence\_of}(s_1,\text{unplug(Dispenser)})) \land \text{min\_precedes}(s,s_1,a)) \land (\exists s_2 \ (\text{occurrence\_of}(s_2,\text{turnon(Dispenser)})) \land \text{min\_precedes}(s_1,s_2,a)))))\].

**Results:**  
Search stopped by max seconds option

Experiment 5.2.22

**Theory:**  
\[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \]

**Query:**  
\[-(\forall s \ (\text{root}(s,\text{sanitize(Dispenser)}) \rightarrow ((\exists s_1 \exists a_1 \ (\text{occurrence\_of}(s_1,a_1) \land \text{min\_precedes}(s,s_1,\text{sanitize(Dispenser)})) \land (\exists s_2 \exists a_2 \ (\text{subactivity}(a_2,a) \land \text{occurrence\_of}(s_2,a_2) \land \text{min\_precedes}(s_1,s_2,\text{sanitize(Dispenser)})))))\].

**Results:**  
Search stopped by max seconds option
Experiment 5.2.23

**Theory:** \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

**Query:**

\[
- (\exists a \exists o \exists s1 \exists s2 \exists a1 \exists a2 \\
\text{activity}(a) \land \text{occurrence}_o(a) \\
\rightarrow \\
(\exists s1 \exists s2 \\
(\text{occurrence}_o(s1, \text{unplug}(\text{Dispenser})) \\
\land \text{subactivity}_o(s1, o)) \\
\land (\exists s2 \exists s1 \\
(\text{occurrence}_o(s2, \text{turnon}(\text{Dispenser})) \\
\land \text{min}_\text{precedes}(s1, s2, a))))).
\]

**Results:** Search stopped by max seconds option

Experiment 5.2.24

**Theory:** \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_2} \)

**Query:**

\[
- (\forall o \exists a1 \exists a2 \\
(\text{occurrence}_o(sanitize(\text{Dispenser}))) \\
\rightarrow \\
(\exists s1 \exists s2 \\
(\text{occurrence}_o(s1, a1) \\
\land \text{subactivity}_o(s1, o)) \\
\land (\exists s2 \exists s1 \\
(\text{subactivity}(a2, a) \\
\land \text{occurrence}_o(s2, a2) \\
\land \text{min}_\text{precedes}(s1, s2, sanitize(\text{Dispenser}))))).
\]

**Results:** Search stopped by max seconds option
Experiment 5.2.25

Theory: $\sum_{pd\_closure} \cup T_{\text{lemma}_2}$

Query: $\neg (\exists a$
    
    (activity(a)
    & (\forall s \forall s1 \forall s2
    ( ( \text{root}(s,a)
    & \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser}))
    & \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser}))
    & \text{min\_precedes}(s,s1,a)
    & \text{min\_precedes}(s,s2,a))
    ->
    (\text{min\_precedes}(s1,s2,a)))))).$

Results: Search stopped by max seconds option

Experiment 5.2.26

Theory: $\sum_{pd\_closure} \cup T_{\text{lemma}_2}$

Query: $\neg (\forall s \forall s1 \forall s2 \forall a1 \forall a2$
    
    ( ( \text{root}(s,\text{sanitize}(\text{Dispenser}))
    & \text{occurrence\_of}(s1,a1)
    & \text{occurrence\_of}(s2,a2)
    & \text{min\_precedes}(s,s1,\text{sanitize}(\text{Dispenser}))
    & \text{min\_precedes}(s,s2,\text{sanitize}(\text{Dispenser}))
    )
    ->
    (\text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})))).$

Results: Search stopped by max seconds option
Experiment 5.2.27

Theory: \[ \sum_{pd\_closure} \cup T_{\text{lemma}_2} \]

Query: \[ \neg(\exists a \quad (\text{activity}(a) \quad \& \quad (\forall o \quad \forall s1 \quad \forall s2 \quad (( \text{occurrence\_of}(o,a) \quad \& \quad \text{occurrence\_of}(s1,\text{unplug(Dispenser)}) \quad \& \quad \text{occurrence\_of}(s2,\text{turnon(Dispenser)}) \quad \& \quad \text{subactivity\_occurrence}(s1,o) \quad \& \quad \text{subactivity\_occurrence}(s2,o) \quad ) \quad \rightarrow \quad (\text{min\_precedes}(s1,s2,a)))))). \]

Results: Search stopped by max seconds option

Experiment 5.2.28

Theory: \[ \sum_{pd\_closure} \cup T_{\text{lemma}_2} \]

Query: \[ \neg(\forall o \quad \forall s1 \quad \forall s2 \quad \forall a1 \quad \forall a2 \quad (( \text{occurrence\_of}(o,\text{sanitize(Dispenser)}) \quad \& \quad \text{occurrence\_of}(s1,a1) \quad \& \quad \text{occurrence\_of}(s2,a2) \quad \& \quad \text{subactivity\_occurrence}(s1,o) \quad \& \quad \text{subactivity\_occurrence}(s2,o) \quad ) \quad \rightarrow \quad (\text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)})))). \]

Results: Search stopped by max seconds option
Experiment 5.2.29

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_2} \]

Query: \[-(exists\ a \\
(\ activity(a) \\
& (all\ s\ all\ s1 \\
((\ root(s,a) \\
& occurrence\_of(s1,unplug(Dispenser)) \\
& min\_precedes(s,s1,a)) \\
-&>

( - (exists\ s2 \\
(occurrence\_of(s2,turnon(Dispenser)) \\
& min\_precedes(s1,s2,a))))))))).

Results: Search stopped by max seconds option

Experiment 5.2.30

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_2} \]

Query: \[-(all\ s\ all\ s1\ all\ a1 \\
((\ root(s,sanitize(Dispenser)) \\
& occurrence\_of(s1,a1) \\
& min\_precedes(s,s1,sanitize(Dispenser))) \\
-&>

( - (exists\ s2\ exists\ a2 \\
(subactivity(a2,a) \\
& occurrence\_of(s2,a2) \\
& min\_precedes(s1,s2,sanitize(Dispenser)) \\
))))).

Results: Search stopped by max seconds option
Experiment 5.2.31

Theory: \( \sum_{pd\_closure} \cup T_{\text{lemma}2} \)

Query: \(-\exists a \left( \text{activity}(a) \right. \\
\left. & \left( \forall o \left( \forall s1 \left( \\
\left( \text{occurrence\_of}(o,a) \\
& \text{occurrence\_of}(s1,\text{unplug(Dispenser)}) \\
& \text{subactivity\_occurrence}(s1,o) \\
\right) \rightarrow \\
\left( -\exists s2 \left( \\
\left( \text{occurrence\_of}(s2,\text{turnon(Dispenser)}) \\
& \text{min\_precedes}(s1,s2,a) \right) \right) \right) \right) \) \).

Results: Search stopped by max seconds option

Experiment 5.2.32

Theory: \( \sum_{pd\_closure} \cup T_{\text{lemma}2} \)

Query: \(-\forall o \forall s1 \forall a1 \left( \left( \text{occurrence\_of}(o,\text{sanitize(Dispenser)}) \\
& \text{occurrence\_of}(s1,a1) \\
& \text{subactivity\_occurrence}(s1,o) \\
\right) \rightarrow \\
\left( -\exists s2 \exists a2 \left( \\
\left( \text{subactivity}(a2,a) \\
& \text{occurrence\_of}(s2,a2) \\
& \text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)}) \right) \right) \right) \).

Results: Search stopped by max seconds option
Experiment 5.2.33

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_2} \]

Query: \[ \neg (\exists a \ (\text{activity}(a) \& (\forall s \ (\text{root}(s,a) \rightarrow \ ((\exists s_1 \ (\text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser})) \& \text{min\_precedes}(s,s_1,a)) \& (\exists s_2 \ (\text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser})) \& \text{min\_precedes}(s_1,s_2,a))))))) \).

Results: Search stopped by max seconds option

Experiment 5.2.34

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_2} \]

Query: \[ \neg (\forall s \ (\text{root}(s,\text{sanitize}(\text{Dispenser})) \rightarrow \ ((\exists s_1 \exists a_1 \ (\text{occurrence\_of}(s_1,a_1) \& \text{min\_precedes}(s,s_1,\text{sanitize}(\text{Dispenser})))) \& (\exists s_2 \exists a_2 \ (\text{subactivity}(a_2,a) \& \text{occurrence\_of}(s_2,a_2) \& \text{min\_precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser})))))))) \).

Results: Search stopped by max seconds option
Experiment 5.2.35

Theory: \( \sum_{pd\_closure} \cup T_{lemma\_2} \)

Query: \(-\left(\exists a \right.
\left.\left(\text{activity}(a) \land \left(\forall o \left(\text{occurrence\_of}(o,a) \Rightarrow \left(\exists s1 \left(\text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \land \text{subactivity\_occurrence}(s1,o)\right) \land \left(\exists s2 \left(\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s1,s2,a)\right)\right)\right)\right)\right)\).

Results: Search stopped by max seconds option

Experiment 5.2.36

Theory: \( \sum_{pd\_closure} \cup T_{lemma\_2} \)

Query: \(-\left(\forall o \left(\text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \Rightarrow \left(\exists s1 \left(\exists a1 \left(\text{occurrence\_of}(s1,a1) \land \text{subactivity\_occurrence}(s1,o)\right) \land \left(\exists s2 \left(\exists a2 \left(\text{subactivity}(a2,a) \land \text{occurrence\_of}(s2,a2) \land \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser}))\right)\right)\right)\right)\right)\).

Results: Search stopped by max seconds option
Experiments 5.3

The following table provides an immediate outline identifying what background theories and form of process description each query is associated with and the corresponding experiment number. Below are the experiment summaries for Experiments 5.3.

Table 5.3. Outline of Experiments 5.3

<table>
<thead>
<tr>
<th>Query</th>
<th>$T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd_closure} \cup T_{\text{lemma}_3}$</th>
<th>$T_{\text{complex}} \cup \sum_{pd_closure} \cup T_{\text{lemma}_3}$</th>
<th>$\sum_{pd_closure} \cup T_{\text{lemma}_3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>5.3.1</td>
<td>5.3.16</td>
<td>5.3.31</td>
</tr>
<tr>
<td>1b</td>
<td>5.3.2</td>
<td>5.3.17</td>
<td>5.3.32</td>
</tr>
<tr>
<td>1c</td>
<td>5.3.3</td>
<td>5.3.18</td>
<td>5.3.33</td>
</tr>
<tr>
<td>2a</td>
<td>5.3.4</td>
<td>5.3.19</td>
<td>5.3.34</td>
</tr>
<tr>
<td>2b</td>
<td>5.3.5</td>
<td>5.3.20</td>
<td>5.3.35</td>
</tr>
<tr>
<td>2c</td>
<td>5.3.6</td>
<td>5.3.21</td>
<td>5.3.36</td>
</tr>
<tr>
<td>2d</td>
<td>5.3.7</td>
<td>5.3.22</td>
<td>5.3.37</td>
</tr>
<tr>
<td>3a</td>
<td>5.3.8</td>
<td>5.3.23</td>
<td>5.3.38</td>
</tr>
<tr>
<td>3b</td>
<td>5.3.9</td>
<td>5.3.24</td>
<td>5.3.39</td>
</tr>
<tr>
<td>3c</td>
<td>5.3.10</td>
<td>5.3.25</td>
<td>5.3.40</td>
</tr>
<tr>
<td>3d</td>
<td>5.3.11</td>
<td>5.3.26</td>
<td>5.3.41</td>
</tr>
<tr>
<td>4a</td>
<td>5.3.12</td>
<td>5.3.27</td>
<td>5.3.42</td>
</tr>
<tr>
<td>4b</td>
<td>5.3.13</td>
<td>5.3.28</td>
<td>5.3.43</td>
</tr>
<tr>
<td>4c</td>
<td>5.3.14</td>
<td>5.3.29</td>
<td>5.3.44</td>
</tr>
<tr>
<td>4d</td>
<td>5.3.15</td>
<td>5.3.30</td>
<td>5.3.45</td>
</tr>
</tbody>
</table>
Experiment 5.3.1

**Theory:** $T_{\text{actocc} \cup \text{complex} \cup \sum_{pd\ _\ closure} \cup T_{\text{lemma\_3}}}$

**Query:** $\neg (\exists a \ (\text{activity}(a) \ & \ (\forall s \ \ (\text{root}(s,a) \ \rightarrow \ \ (\exists s1 \ \exists s2 \ \exists s \ \ (\text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \ & \ \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \ & \ \text{min\_precedes}(s,s1,a) \ & \ \text{min\_precedes}(s,s2,a) \ & \ \text{min\_precedes}(s1,s2,a) \ & \ \$\text{ans}(a)))))) \}.$

**Results:** Search stopped by max seconds option

Experiment 5.3.2

**Theory:** $T_{\text{actocc} \cup \text{complex} \cup \sum_{pd\ _\ closure} \cup T_{\text{lemma\_3}}}$

**Query:** $\neg (\forall s \ \ (\text{root}(s,\text{sanitize}(\text{Dispenser})) \ \rightarrow \ \ (\exists s1 \ \exists s2 \ \exists s \ \ (\text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \ & \ \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \ & \ \text{min\_precedes}(s,s1,a) \ & \ \text{min\_precedes}(s,s2,a) \ & \ \text{min\_precedes}(s1,s2,a) \ & \ \$\text{ans}(a)))) \}.$

**Results:** Search stopped by max seconds option
Experiment 5.3.3

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_closure}} \cup T_{\text{lemma\_3}} \)

Query: \(- (\exists a \left( \text{activity}(a) \land (\forall o \left( \text{occurrence\_of}(o, a) \implies \right. \right.
\left. \exists s_1 \exists s_2 \exists s \left( \text{occurrence\_of}(s_1, \text{unplug}(\text{Dispenser})) \land \text{occurrence\_of}(s_2, \text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s, s_1, a) \land \text{min\_precedes}(s, s_2, a) \land \text{min\_precedes}(s_1, s_2, a) \land \$\text{ans}(a)) \right) ) \).\)

Results: Proof Generated: \(-\$\text{ans}(\text{drain}(\text{Water, Dispenser}))\)
See Appendix B, Figure B1 for details of the proof.
Duration: 87.27 sec

Experiment 5.3.4

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_closure}} \cup T_{\text{lemma\_3}} \)

Query: \(- (\exists a \left( \text{activity}(a) \land (\forall s \forall s_1 \forall s_2 \left( ( \text{root}(s, a) \land \text{occurrence\_of}(s_1, \text{unplug}(\text{Dispenser})) \land \text{occurrence\_of}(s_2, \text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s, s_1, a) \land \text{min\_precedes}(s, s_2, a) \land \$\text{ans}(a) \right) \implies \right. \right.
\left. \text{min\_precedes}(s_1, s_2, a) \right) ) \).\)

Results: Proof Generated: \$\text{ans}(\text{sanitize}(\text{Dispenser}))\)
See Appendix B, Figure B2 for details of the proof.
Duration: 0.09 sec
Experiment 5.3.5

Theory: \( T_{actocc} \cup \text{complex} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

Query: \(-\left(\text{all } s \text{ all } s1 \text{ all } s2 \left(\begin{array}{l}
\left( \text{root}(s, \text{drain(Water,Dispenser)})
\& \text{occurrence}_\text{of}(s1, \text{unplug(Dispenser)})
\& \text{occurrence}_\text{of}(s2, \text{turnon(Dispenser)})
\& \text{min}_\text{precedes}(s, s1, \text{drain(Water,Dispenser)})
\& \text{min}_\text{precedes}(s, s2, \text{drain(Water,Dispenser)})
\end{array}\right)
\implies
\left(\text{min}_\text{precedes}(s1, s2, \text{drain(Water,Dispenser)})\right)\right)\).

Results: Search stopped by max seconds option

Experiment 5.3.6

Theory: \( T_{actocc} \cup \text{complex} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

Query: \(-\left(\text{exists } a \left(\begin{array}{l}
\text{activity}(a)
\& \left(\text{all } o \text{ all } s1 \text{ all } s2 \left(\begin{array}{l}
\text{occurrence}_\text{of}(o, a)
\& \text{occurrence}_\text{of}(s1, \text{unplug(Dispenser)})
\& \text{occurrence}_\text{of}(s2, \text{turnon(Dispenser)})
\& \text{subactivity}_\text{occurrence}(s1, o)
\& \text{subactivity}_\text{occurrence}(s2, o)
\& \$\text{ans}(a))
\end{array}\right)
\implies
\left(\text{min}_\text{precedes}(s1, s2, a)\right)\right)\).

Results: Proof Generated: \$\text{ans}(\text{sanitize(Dispenser)})$

See Appendix B, Figure B3 for details of the proof.

Duration: 0.13sec
Experiment 5.3.7

Theory: \[ T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pdclosure}} \cup T_{\text{lemma}_3} \]

Query: \[-(\text{all } o \text{ all } s1 \text{ all } s2
  \quad (\text{occurrence_of}(o, \text{drain(Water,Dispenser)})
  \quad \& \text{occurrence_of}(s1, \text{unplug(Dispenser)})
  \quad \& \text{occurrence_of}(s2, \text{turnon(Dispenser)})
  \quad \& \text{subactivity_occurrence}(s1,o)
  \quad \& \text{subactivity_occurrence}(s2,o))
  \quad \rightarrow
  \quad \text{(min_precedes}(s1,s2, \text{drain(Water,Dispenser)))))).]\\

Results: Search stopped by max seconds option

Experiment 5.3.8

Theory: \[ T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pdclosure}} \cup T_{\text{lemma}_3} \]

Query: \[-(\text{exists } a
  \quad (\text{activity}(a)
  \quad \& \text{(all } s \text{ all } s1
  \quad (\text{root}(s,a)
  \quad \& \text{occurrence_of}(s1, \text{unplug(Dispenser)})
  \quad \& \text{min_precedes}(s,s1,a))
  \quad \rightarrow
  \quad ( - (\text{exists } s2
  \quad (\text{occurrence_of}(s2, \text{turnon(Dispenser)})
  \quad \& \text{min_precedes}(s1,s2,a)
  \quad \& \text{ans}(a))))))).]\\

Results: Proof Generated: \$\text{ans(sanitize(Dispenser))}\$

See Appendix B, Figure B4 for details of the proof

Duration: 0.08sec
Experiment 5.3.9

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd~closure}} \cup T_{\text{lemma~3}} \)

Query: \(-\text{all s all s1}
((-\text{root(s,drain(Water,Dispenser))})
& \text{occurrence_of(s1,unplug(Dispenser))}
& \text{min_precedes(s,s1,drain(Water,Dispenser)))}
\rightarrow
( - (\text{exists s2})
(\text{occurrence_of(s2,turnon(Dispenser))}
& \text{min_precedes(s1,s2,drain(Water,Dispenser))))
))\).

Results: Search stopped by max seconds option

Experiment 5.3.10

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd~closure}} \cup T_{\text{lemma~3}} \)

Query: \(-\text{exists a}
(\text{activity(a)})
& (\text{all o all s1})
((-\text{occurrence_of(o,a)})
& \text{occurrence_of(s1,unplug(Dispenser))}
& \text{subactivity_occurrence(s1,o)})
\rightarrow
( - (\text{exists s2})
(\text{occurrence_of(s2,turnon(Dispenser))}
& \text{min_precedes(s1,s2,a)}
& $\text{ans(a))})()))).

Results: Proof Generated: \$\text{ans(sanitize(Dispenser))\
See Appendix B, Figure B5 for details of the proof.

Duration: 0.16sec
Experiment 5.3.11

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_cloure}} \cup T_{\text{lemma\_3}} \)

Query: \(-\forall o \; \forall s1 \; \forall a1 \;
(\exists o \; (\text{occurrence\_of}(o, \text{sanitize}(\text{Dispenser})) \\
\quad \& \; (\text{occurrence\_of}(s1,a1) \\
\quad \& \; (\text{subactivity\_occurrence}(s1,o))) \\
-\rightarrow \\
\quad (\exists s2 \; \exists a2 \\
\quad (\text{subactivity}(a2,a) \\
\quad \& \; (\text{occurrence\_of}(s2,a2) \\
\quad \& \; (\text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \\
\quad \& \; (\$\text{ans}(a1)) \\
\quad \& \; (\$\text{ans}(a2)))))))).\)

Results: Search stopped by max seconds options

Experiment 5.3.12

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{\text{pd\_cloure}} \cup T_{\text{lemma\_3}} \)

Query: \(-\exists a \\
(\forall s \\
(\text{root}(s,a)) \\
-\rightarrow \\
(\exists s1 \\
(\text{occurrence\_of}(s1, \text{unplug}(\text{Dispenser})) \\
\quad \& \; (\text{min\_precedes}(s,s1,a))) \\
\& (\exists s2 \\
(\text{occurrence\_of}(s2, \text{turnon}(\text{Dispenser})) \\
\quad \& \; (\text{min\_precedes}(s1,s2,a) \\
\quad \& \; (\$\text{ans}(a))))))).\)

Results: Search stopped by max seconds option
Experiment 5.3.13

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_closures} \cup T_{\text{lemma}\_3} \)

Query: \(- (\forall s \\
(\text{root}(s,\text{sanitize}(\text{Dispenser}))) \\
\rightarrow \\
((\exists s_1 \exists a_1 \\
(\text{occurrence}\_of(s_1,a_1) \\
& \text{min}\_\text{precedes}(s,s_1,\text{sanitize}(\text{Dispenser})))) \\
& (\exists s_2 \exists a_2 \\
(\text{subactivity}(a_2,a) \\
& \text{occurrence}\_of(s_2,a_2) \\
& \text{min}\_\text{precedes}(s_1,s_2,\text{sanitize}(\text{Dispenser})) \\
& \$\text{ans}(a_1) \\
& \$\text{ans}(a_2)))\)).\)

Results: Search stopped by max seconds option

Experiment 5.3.14

Theory: \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_closures} \cup T_{\text{lemma}\_3} \)

Query: \(- (\exists a \\
(\text{activity}(a) \\
& (\forall o \\
(\text{occurrence}\_of(o,a) \\
\rightarrow \\
((\exists s_1 \\
(\text{occurrence}\_of(s_1,\text{unplug}(\text{Dispenser})) \\
& \text{subactivity}\_\text{occurrence}(s_1,o)) \\
& (\exists s_2 \\
(\text{occurrence}\_of(s_2,\text{turnon}(\text{Dispenser})) \\
& \text{min}\_\text{precedes}(s_1,s_2,a) \\
& \$\text{ans}(a))))))\)).\)

Results: Search stopped by max seconds option
Experiment 5.3.15

**Theory:** \( T_{\text{actocc}} \cup \text{complex} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

**Query:**
\[
\neg (\forall o \exists s1 \exists a1 \exists s2 \exists a2 \\
(\text{occurrence}\_\text{of}(o,\text{sanitize}(\text{Dispenser}))) \\
\Rightarrow \\
(\text{occurrence}\_\text{of}(s1,a1) \\
\& \text{subactivity}\_\text{occurrence}(s1,o)) \)
\& \\
(\text{subactivity}(a2,a) \\
\& \text{occurrence}\_\text{of}(s2,a2) \)
\& \text{min}\_\text{precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \)
\& \text{\$ans(a1)} \)
\& \text{\$ans(a2)})))).
\]

**Results:** Search stopped by max seconds option

Experiment 5.3.16

**Theory:** \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

**Query:**
\[
\neg (\exists a \\
(\text{activity}(a) \\
\& (\forall s \\
(\text{root}(s,a) \\
\Rightarrow \\
(\text{occurrence}\_\text{of}(s1,\text{unplug}(\text{Dispenser})) \\
\& \text{occurrence}\_\text{of}(s2,\text{turnon}(\text{Dispenser})) \\
\& \text{min}\_\text{precedes}(s,s1,a) \\
\& \text{min}\_\text{precedes}(s,s2,a) \\
\& \text{min}\_\text{precedes}(s1,s2,a) \)
\& \text{\$ans(a)}))).
\]

**Results:** Search stopped by max seconds option
Experiment 5.3.17

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma\_3}} \]

Query: \[-(\text{all } s \\
\quad (\text{root}(s, \text{sanitize(Dispenser)})) \\
\quad \rightarrow \\
\quad (\exists s1 \exists s2 \exists s \\
\quad (\text{occurrence\_of}(s1, \text{unplug(Dispenser)}) \\
\quad \quad \& \quad \text{occurrence\_of}(s2, \text{turnon(Dispenser)}) \\
\quad \quad \& \quad \text{min\_precedes}(s,s1,a) \\
\quad \quad \& \quad \text{min\_precedes}(s,s2,a) \\
\quad \quad \& \quad \text{min\_precedes}(s1,s2,a) \\
\quad \quad \& \quad \$\text{ans}(a))))).\]

Results: Search stopped by max seconds option

Experiment 5.3.18

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma\_3}} \]

Query: \[-(\exists a \\
\quad (\text{activity}(a) \\
\quad \& \quad (\text{all } o \\
\quad \quad (\text{occurrence\_of}(o,a)) \\
\quad \quad \rightarrow \\
\quad \quad (\exists s1 \exists s2 \exists s \\
\quad \quad (\text{occurrence\_of}(s1, \text{unplug(Dispenser)}) \\
\quad \quad \& \quad \text{occurrence\_of}(s2, \text{turnon(Dispenser)}) \\
\quad \quad \& \quad \text{min\_precedes}(s,s1,a) \\
\quad \quad \& \quad \text{min\_precedes}(s,s2,a) \\
\quad \quad \& \quad \text{min\_precedes}(s1,s2,a) \\
\quad \quad \& \quad \$\text{ans}(a)))\)).\]

Results: Proof: \(-\$\text{ans}(\text{drain(Water,Dispenser)})\)
See Appendix B, Figure B6 for details of the proof.
Duration: 21.05sec
Experiment 5.3.19

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma\_3}} \]

Query: \[-(\exists a \)
\hspace{1em} (activity(a)
\hspace{1em} & (\forall s \forall s1 \forall s2
\hspace{1em} (\text{root}(s,a)
\hspace{1em} & \text{occurrence\_of}(s1,unplug(\text{Dispenser}))
\hspace{1em} & \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser}))
\hspace{1em} & \text{min\_precedes}(s,s1,a)
\hspace{1em} & \text{min\_precedes}(s,s2,a)
\hspace{1em} & \$\text{ans}(a))
\hspace{1em} ->
\hspace{1em} (\text{min\_precedes}(s1,s2,a))))).
\]

Results: Proof Generated: \$\text{ans}(\text{sanitize}(\text{Dispenser}))$

See Appendix B, Figure B7 for details of the proof.

Duration: 0.08sec

Experiment 5.3.20

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma\_3}} \]

Query: \[-(\forall s \forall s1 \forall s2
\hspace{1em} ((\text{root}(s,\text{drain}(\text{Water},\text{Dispenser}))
\hspace{1em} & \text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser}))
\hspace{1em} & \text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser}))
\hspace{1em} & \text{min\_precedes}(s,s1,\text{drain}(\text{Water},\text{Dispenser}))
\hspace{1em} & \text{min\_precedes}(s,s2,\text{drain}(\text{Water},\text{Dispenser}))
\hspace{1em} ->
\hspace{1em} (\text{min\_precedes}(s1,s2,\text{drain}(\text{Water},\text{Dispenser})))).
\]

Results: Search stopped by max seconds option
Experiment 5.3.21

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

Query: \(-\exists a \ (\text{activity}(a) \land \forall o \forall s_1 \forall s_2 \ ((\text{occurrence}_o(o,a) \land \text{occurrence}_o(s_1,\text{unplug}(\text{Dispenser})) \land \text{occurrence}_o(s_2,\text{turnon}(\text{Dispenser})) \land \text{subactivity}_o(s_1,o) \land \text{subactivity}_o(s_2,o) \land \text{ans}(a)) \rightarrow (\text{min}_o\text{precedes}(s_1,s_2,a)))).\)

Results: Proof Generated: \$\text{ans}(\text{sanitize}(\text{Dispenser}))$

See Appendix B, Figure B8 for details of the proof.

Duration: 0.11 sec

Experiment 5.3.22

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \)

Query: \(-\forall o \forall s_1 \forall s_2 \ ((\text{occurrence}_o(o,\text{drain}(\text{Water},\text{Dispenser})) \land \text{occurrence}_o(s_1,\text{unplug}(\text{Dispenser})) \land \text{occurrence}_o(s_2,\text{turnon}(\text{Dispenser})) \land \text{subactivity}_o(s_1,o) \land \text{subactivity}_o(s_2,o)) \rightarrow (\text{min}_o\text{precedes}(s_1,s_2,\text{drain}(\text{Water},\text{Dispenser}))))).\)

Results: Search stopped by max seconds option
Experiment 5.3.23

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \]

Query: \[-(\exists a \neg (\text{activity}(a) \land (\forall s \forall s1 (\text{root}(s,a) \land \text{occurrence}_o(s1,\text{unplug}(\text{Dispenser})) \land \text{min\_precedes}(s,s1,a) \Rightarrow \neg (\exists s2 (\text{occurrence}_o(s2,\text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s1,s2,a) \land \text{ans}(a))))))). \]

Results: Proof Generated: \$\text{ans}(\text{sanitize}(\text{Dispenser}))$

See Appendix B, Figure B9 for details of the proof.

Duration: 0.03sec

Experiment 5.3.24

Theory: \[ T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}_3} \]

Query: \[-(\forall s \forall s1 ((\text{root}(s,\text{drain}(\text{Water},\text{Dispenser})) \land \text{occurrence}_o(s1,\text{unplug}(\text{Dispenser})) \land \text{min\_precedes}(s,s1,\text{drain}(\text{Water},\text{Dispenser})) \Rightarrow \neg (\exists s2 (\text{occurrence}_o(s2,\text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s1,s2,\text{drain}(\text{Water},\text{Dispenser}))))) \])\]

Results: Search stopped by max seconds option
Experiment 5.3.25

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}\_3} \)

Query: \(-\text{(exists a} \\ (\text{activity}(a) \\ \\
\& \ (all \ o \ all \ s1 \ \\
\text{(( occurrence_of(o,a) \ \\
\& \ occurrence_of(s1,unplug(\text{Dispenser})) \ \\
\& \ subactivity\_occurrence(s1,o)) \ \\
\text{->} \ \\
\text{((exists s2} \ \\
\text{occurrence_of(s2,turnon(\text{Dispenser})) \ \\
\& \ min\_precedes(s1,s2,a) \ \\
\& \ \text{\$ans(a1)})))))\)).

Results: Proof Generated : \text{\$ans(sanitize(\text{Dispenser}))}
See Appendix B, Figure B10 for details of the proof.
Duration: 0.03sec

Experiment 5.3.26

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}\_3} \)

Query: \(-\text{(all o all s1 all a1} \ \\
\text{(( occurrence_of(o,sanitize(\text{Dispenser})) \ \\
\& \ occurrence_of(s1,a1) \ \\
\& \ subactivity\_occurrence(s1,o)) \ \\
\text{->} \ \\
\text{((exists s2 exists a2} \ \\
\text{subactivity(a2,a) \ \\
\& \ occurrence_of(s2,a2) \ \\
\& \ min\_precedes(s1,s2,sanitize(\text{Dispenser})) \ \\
\& \ \text{\$ans(a1)} \ \\
\& \ \text{\$ans(a2)))))\)).

Results: Search stopped by max seconds options
Experiment 5.3.27

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}\_3} \)

Query: \( \neg(\exists a \ (\text{activity}(a) \land (\forall s \ (\text{root}(s,a) \implies ((\exists s_1 \ (\text{occurrence\_of}(s_1, \text{unplug}(\text{Dispenser})) \land \text{min\_precedes}(s,s_1,a)) \land (\exists s_2 \ (\text{occurrence\_of}(s_2, \text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s_1,s_2,a) \land \$\text{ans}(a))))))). \)

Results: Search stopped by max seconds option

Experiment 5.3.28

Theory: \( T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}\_3} \)

Query: \( \neg(\forall s \ (\text{root}(s, \text{sanitize}(\text{Dispenser})) \implies ((\exists a_1 \exists s_1 \ (\text{occurrence\_of}(s_1, a_1) \land \text{min\_precedes}(s,s_1,a) \land (\exists a_2 \exists s_2 \ (\text{subactivity}(a_2,a) \land \text{occurrence\_of}(s_2, a_2) \land \text{min\_precedes}(s_1,s_2,a) \land \$\text{ans}(a_1) \land \$\text{ans}(a_2))))))). \)

Results: Search stopped by max seconds option
Experiment 5.3.29

**Theory:** $T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}3}$

**Query:** $- \,(\exists \, a \, (\text{activity}(a) \& (\forall \, o \, (\text{occurrence\_of}(o,a) \rightarrow \left( (\exists \, s1 \, (\text{occurrence\_of}(s1,\text{unplug(Dispenser)}) \& \text{subactivity\_occurrence}(s1,o)) \& (\exists \, s2 \, (\text{occurrence\_of}(s2,\text{turnon(Dispenser)}) \& \text{min\_precedes}(s1,s2,a) \& \$\text{ans}(a)))))))))$.

**Results:** Search stopped by max seconds option

Experiment 5.3.30

**Theory:** $T_{\text{complex}} \cup \sum_{pd\_closure} \cup T_{\text{lemma}3}$

**Query:** $- (\forall \, o \, (\text{occurrence\_of}(o,\text{sanitize(Dispenser)}) \rightarrow \left( (\exists \, s1 \, \exists \, a1 \, (\text{occurrence\_of}(s1,a1) \& \text{subactivity\_occurrence}(s1,o)) \& (\exists \, s2 \, \exists \, a2 \, (\text{subactivity}(a2,a) \& \text{occurrence\_of}(s2,a2) \& \text{min\_precedes}(s1,s2,\text{sanitize(Dispenser)}) \& \$\text{ans}(a1) \& \$\text{ans}(a2))))))$.

**Results:** Search stopped by max seconds option
Experiment 5.3.31

Theory: $\sum_{pd\_cloure} \cup T_{\text{lemma}_3}$

Query: $-(\exists a$
\hspace{1em} (\text{activity}(a) \\
\hspace{2em} \& (\forall s$
\hspace{3em} (\text{root}(s,a) \\
\hspace{4em} \rightarrow$
\hspace{5em} (\exists s_1 \exists s_2 \exists s$
\hspace{6em} (\text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser})) \\
\hspace{7em} \& \text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser})) \\
\hspace{8em} \& \text{min\_precedes}(s,s_1,a) \\
\hspace{9em} \& \text{min\_precedes}(s,s_2,a) \\
\hspace{10em} \& \text{min\_precedes}(s_1,s_2,a) \\
\hspace{11em} \& \$\text{ans}(a)))\)))).$

Results: Proof: $-\$\text{ans}(\text{drain}(\text{Water},\text{Dispenser}))$

See Appendix B, Figure B11 for details of the proof.

Duration: 122.34sec

Experiment 5.3.32

Theory: $\sum_{pd\_cloure} \cup T_{\text{lemma}_3}$

Query: $-(\forall s$
\hspace{1em} (\text{root}(s,\text{sanitize}(\text{Dispenser})) \\
\hspace{2em} \rightarrow$
\hspace{3em} (\exists s_1 \exists s_2 \exists s$
\hspace{4em} (\text{occurrence\_of}(s_1,\text{unplug}(\text{Dispenser})) \\
\hspace{5em} \& \text{occurrence\_of}(s_2,\text{turnon}(\text{Dispenser})) \\
\hspace{6em} \& \text{min\_precedes}(s,s_1,a) \\
\hspace{7em} \& \text{min\_precedes}(s,s_2,a) \\
\hspace{8em} \& \text{min\_precedes}(s_1,s_2,a) \\
\hspace{9em} \& \$\text{ans}(a)))\))$.}

Results: Search stopped by max seconds option
Experiment 5.3.33

Theory: \[ \sum_{pd \_ cloure} \cup T_{lemma \_ 3} \]

Query: \[ -(exists \ a \\
\quad (activity(a) \\
\quad & (all \ o \\
\quad (occurrence \_ of(o,a)) \\
\quad -> \\
\quad (exists \ s1 \ exists \ s2 \ exists \ s \\
\quad (occurrence \_ of(s1,unplug(Dispenser)) \\
\quad & occurrence \_ of(s2,turnon(Dispenser)) \\
\quad & min \_ precedes(s,s1,a) \\
\quad & min \_ precedes(s,s2,a) \\
\quad & min \_ precedes(s1,s2,a) \\
\quad & $ans(a)))))). \]

Results: Proof: \(-$ans(drain(Water,Dispenser))\)
See Appendix B, Figure B12 for details of the proof.
Duration: 1.48sec

Experiment 5.3.34

Theory: \[ \sum_{pd \_ cloure} \cup T_{lemma \_ 3} \]

Query: \[ -(exists \ a \\
\quad (activity(a) \\
\quad & (all \ s \ all \ s1 \ all \ s2 \\
\quad ( root(s,a) \\
\quad & occurrence \_ of(s1,unplug(Dispenser)) \\
\quad & occurrence \_ of(s2,turnon(Dispenser)) \\
\quad & min \_ precedes(s,s1,a) \\
\quad & min \_ precedes(s,s2,a) \\
\quad & $ans(a)) \\
\quad -> \\
\quad (min \_ precedes(s1,s2,a)))))). \]

Results: Proof Generated: \$ans(sanitize(Dispenser))\)
See Appendix B, Figure B13 for details of the proof.
Duration: 0.01sec
Experiment 5.3.35

Theory: \( \sum_{pd \ _\ closure} \cup T_{lemma \_3} \)

Query: \(- (all \ s \ all \ s1 \ all \ s2
\ ( ( root(s,drain(Water,Dispenser))
& occurrence_of(s1,unplug(Dispenser))
& occurrence_of(s2,turnon(Dispenser))
& min\_precedes(s,s1,drain(Water,Dispenser))
& min\_precedes(s,s2,drain(Water,Dispenser)))
\rightarrow
(min\_precedes(s1,s2,drain(Water,Dispenser))))).\)

Results: Search stopped by max seconds option

Experiment 5.3.36

Theory: \( \sum_{pd \ _\ closure} \cup T_{lemma \_3} \)

Query: \(- (exists \ a
\ (activity(a)
& (all \ o \ all \ s1 \ all \ s2
\ ( ( occurrence_of(o,a)
& occurrence_of(s1,unplug(Dispenser))
& occurrence_of(s2,turnon(Dispenser))
& subactivity\_occurrence(s1,o)
& subactivity\_occurrence(s2,o)
& \$ans(a))
\rightarrow
(min\_precedes(s1,s2,a)))))).\)

Results: Proof Generated : \$ans(sanitize(Dispenser))
See Appendix B, Figure B14 for details of the proof.
Duration: 0.08sec
Experiment 5.3.37

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_3} \]

Query: \[-(all \ o \ all \ s1 \ all \ s2 \]
\[\ (( \ occurrence\_of(o,drain(Water,Dispenser)) \]
\[& \ occurrence\_of(s1,unplug(Dispenser)) \]
\[& \ occurrence\_of(s2,turnon(Dispenser)) \]
\[& \ subactivity\_occurrence(s1,o) \]
\[& \ subactivity\_occurrence(s2,o) \]
\[-> \]
\[\ (min\_precedes(s1,s2,drain(Water,Dispenser))))}.\]

Results: Search stopped by max seconds option

Experiment 5.3.38

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_3} \]

Query: \[-(exists \ a \]
\[\ (activity(a) \]
\[& \ (all \ s \ all \ s1 \]
\[\ (( \ root(s,a) \]
\[& \ occurrence\_of(s1,unplug(Dispenser)) \]
\[& \ min\_precedes(s,s1,a) \]
\[-> \]
\[\ (- \ (exists \ s2 \]
\[\ (occurrence\_of(s2,turnon(Dispenser)) \]
\[& \ min\_precedes(s1,s2,a) \]
\[& \ ans(a))))))))\].

Results: Proof Generated: $ans(sanitize(Dispenser))$
See Appendix B, Figure B15 for details of the proof.
Duration: 0.01sec
Experiment 5.3.39

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_3} \]

Query: \[-(\text{all } s \text{ all } s1 \\
( ( \text{root}(s, \text{drain}(\text{Water, Dispenser})) \\
& \text{occurrence\_of}(s1, \text{unplug}(\text{Dispenser})) \\
& \text{min\_precedes}(s, s1, \text{drain}(\text{Water, Dispenser}))) \\
\rightarrow \\
( - (\text{exists } s2 \\
(\text{occurrence\_of}(s2, \text{turnon}(\text{Dispenser})) \\
& \text{min\_precedes}(s1, s2, \text{drain}(\text{Water, Dispenser})))) \\
)))).\]

Results: Search stopped by max seconds option

Experiment 5.3.40

Theory: \[ \sum_{pd\_closure} \cup T_{lemma\_3} \]

Query: 
- (exists a \\
( \text{activity}(a) \\
& (\text{all } o \text{ all } s1 \\
( ( \text{occurrence\_of}(o, a) \\
& \text{occurrence\_of}(s1, \text{unplug}(\text{Dispenser})) \\
& \text{subactivity\_occurrence}(s1, o)) \\
\rightarrow \\
( - (\text{exists } s2 \\
(\text{occurrence\_of}(s2, \text{turnon}(\text{Dispenser})) \\
& \text{min\_precedes}(s1, s2, a) \\
& \$ans(a))))))).

Results: Proof Generated: \$ans(\text{sanitize}(\text{Dispenser}))

See Appendix B, Figure B16 for details of the proof.

Duration: 0.01sec
Experiment 5.3.41

Theory: \( \sum_{pd\_closure} \cup T_{lemma\_3} \)

Query: \(-\left(\forall o \forall s1 \forall a1 \\left(\left(\text{occurrence\_of}(o,\text{sanitize}(\text{Dispenser})) \land \text{occurrence\_of}(s1,a1) \land \text{subactivity\_occurrence}(s1,o)\right) \rightarrow \left(\neg \left(\exists s2 \exists a2 \left(\text{subactivity}(a2,a) \land \text{occurrence\_of}(s2,a2) \land \text{min\_precedes}(s1,s2,\text{sanitize}(\text{Dispenser})) \land \$\text{ans}(a1) \land \$\text{ans}(a2)\right)\right)\right)\right)\).

Results: Search stopped by max seconds options

Experiment 5.3.42

Theory: \( \sum_{pd\_closure} \cup T_{lemma\_3} \)

Query: \(-\left(\exists a \left(\text{activity}(a) \land \left(\forall s \right) \left(\text{root}(s,a) \rightarrow \left(\left(\exists s1 \left(\text{occurrence\_of}(s1,\text{unplug}(\text{Dispenser})) \land \text{min\_precedes}(s,s1,a)\right) \land \left(\exists s2 \left(\text{occurrence\_of}(s2,\text{turnon}(\text{Dispenser})) \land \text{min\_precedes}(s1,s2,a) \land \$\text{ans}(a)\right)\right)\right)\right)\right)\).

Results: Search stopped by max seconds option
Experiment 5.3.43

Theory: \[ \sum_{pd\_closure} \cup T_{\text{lemma}_3} \]

Query: \[-(\text{all } s)\]

\[\text{(root}(s, \text{sanitize}(\text{Dispenser})) \]

\[- \rightarrow \]

\[((\text{exists } s_1 \text{ exists } a_1)\]

\[\text{(occurrence\_of}(s_1, a_1)\]

& \[\text{min\_precedes}(s, s_1, \text{sanitize}(\text{Dispenser}))\]

& (exists s_2 exists a_2

(subactivity(a_2, a)

& occurrence\_of(s_2, a_2)

& min\_precedes(s_1, s_2, \text{sanitize}(\text{Dispenser}))

& \$\text{ans}(a_1)\]

& \$\text{ans}(a_2)))\).

Results: Search stopped by max seconds option

Experiment 5.3.44

Theory: \[ \sum_{pd\_closure} \cup T_{\text{lemma}_3} \]

Query: \[-(\text{exists } a)\]

\[\text{(activity}(a)\]

& (\text{all } o

\[\text{(occurrence\_of}(o, a)\]

\[- \rightarrow \]

\[((\text{exists } s_1\]

\[\text{(occurrence\_of}(s_1, \text{unplug}(\text{Dispenser}))\]

& \text{subactivity\_occurrence}(s_1, o))\]

& (exists s_2

\[\text{(occurrence\_of}(s_2, \text{turnon}(\text{Dispenser}))\]

& min\_precedes(s_1, s_2, a)

& \$\text{ans}(a))))\).

Results: Search stopped by max seconds option
Experiment 5.3.45

Theory: $\sum_{pd\_cloure} \cup T_{\text{lemma}_3}$

Query: $-(\forall o (\text{occurrence}\_of(o,\text{sanitize(Dispenser)})) \rightarrow (\exists s1 \exists a1 (\text{occurrence}\_of(s1,a1) \& \text{subactivity}\_occurrence(s1,o))) \& (\exists s2 \exists a2 (\text{subactivity}(a2,a) \& \text{occurrence}\_of(s2,a2) \& \text{min}\_precedes(s1,s2,\text{sanitize(Dispenser)}) \& ans(a1) \& ans(a2))))$.

Results: Search stopped by max seconds option
6.0 ONTOLOGY ANALYSIS

This section of the report discusses the lemmas that were used in the experiments. It also identifies the incompleteness in the PSL ontology, discovered over the course of the project, and provides a manual proof of the Transitivity axiom.

A series of hints (i.e. the lemmas) were used in the experiments to aid the theorem prover in the reasoning process. The purpose of including the lemmas as additional theories is to support the hypothesis that by providing the theorem prover with axioms that it would derive along the course of its reasoning, it will reduce the computational time it takes to generate a proof. The lemmas identified for use in the experiments are axioms that are required to generate proofs in the experiments. The composition of the lemmas was discussed in Section 4.2 of the report. It is important to note that Otter is unable to generate a proof for T_lemma_2, and T_lemma_3, although a proof by hand does exist. Otter was able to generate a proof to T_lemma_1, the result of which can be found in Appendix C, Figure C1.

After investigating Otter’s inability to generate a proof for the Transitivity Axiom (which when combined with T_lemma_1 composes T_lemma_2) an interesting discovery was made. There exists incompleteness within the PSL ontology which prevented the automated proof of the Transitivity axiom. A corrected proof, performed by hand, is provided below in Figure 6.1.
Lemma 1. The \textit{min\_precedes} relation is transitive for atomic activity occurrences in the same activity tree.

\[
T_{\text{complex}} \cup T_{\text{atomic}} \cup T_{\text{subactivity}} \cup T_{\text{occtree}} \cup T_{\text{pslcore}} \models \\
(\forall a, s_1, s_2, s_3, s_4, s_5) \text{min\_precedes}(s_1, s_2, a) \land \text{min\_precedes}(s_2, s_3, a) \Rightarrow \text{min\_precedes}(s_1, s_3, a)
\]

Proof.

\[
T_{\text{complex}} \cup T_{\text{atomic}} \cup T_{\text{subactivity}} \cup T_{\text{occtree}} \cup T_{\text{pslcore}} \models \\
\text{min\_precedes}(s_1, s_2) \land \text{min\_precedes}(s_2, s_3, a) \Rightarrow \text{precedes}(s_1, s_2) \land \text{precedes}(s_2, s_3)
\]

The transitivity of \textit{precedes} from \textit{T}_{\text{occtree}} gives us

\[
T_{\text{complex}} \cup T_{\text{atomic}} \cup T_{\text{subactivity}} \cup T_{\text{occtree}} \cup T_{\text{pslcore}} \models \\
\text{min\_precedes}(s_1, s_2) \land \text{min\_precedes}(s_2, s_3, a) \Rightarrow \text{precedes}(s_1, s_3)
\]

By Axiom 4 of \textit{T}_{\text{complex}},

\[
T_{\text{complex}} \cup T_{\text{atomic}} \cup T_{\text{subactivity}} \cup T_{\text{occtree}} \cup T_{\text{pslcore}} \models \\
\text{min\_precedes}(s_1, s_2) \land \text{min\_precedes}(s_2, s_3, a) \Rightarrow (\exists s_4) \text{min\_precedes}(s_4, s_1, a) \land \text{min\_precedes}(s_4, s_3, a)
\]

Axiom 10 of \textit{T}_{\text{complex}} gives us

\[
T_{\text{complex}} \cup T_{\text{atomic}} \cup T_{\text{subactivity}} \cup T_{\text{occtree}} \cup T_{\text{pslcore}} \models \\
\text{min\_precedes}(s_1, s_2) \land \text{min\_precedes}(s_2, s_3, a) \Rightarrow \text{min\_precedes}(s_1, s_3, a)
\]

To facilitate a proof of the Transitivity Axiom it was necessary to include the following axiom:

\[
\forall s_1, s_2, s_3, a \\
((\text{min\_precedes}(s_1, s_2, a) \\
& \text{min\_precedes}(s_3, s_2, a)) \\
\rightarrow \\
(\text{min\_precedes}(s_1, s_3, a) \mid \text{min\_precedes}(s_3, s_1, a) \mid (s_1=s_3))).
\]

The above axiom is included in \textit{T}_{\text{lemma\_3}} and used as one of the hints in the experiments performed.
7.0 SUMMARY OF EXPERIMENTS

The following discussion is based on the experimental results presented in Section 5.0 of the report.

Throughout experiments 5.1.1-5.1.24 $T_{\text{lemma}_1}$ was consistently included while the background ontologies were varied. Experiments 5.1.1-5.1.12 included $T_{\text{actocc}} \cup \text{complex}$, experiments 5.1.13-5.1.24 included $T_{\text{complex}}$, and experiments 5.1.25-5.1.36 only included $\sum_{pd_{\text{complex}}}$ and $T_{\text{lemma}_2}$. Otter was unable to generate a proof to any of the queries posed to the system. Otter ‘timed-out’ on generating a proof for these queries as the system was limited to 1800secs (30mins.) to generate a proof. A potential reason for this inability to generate proofs was initially thought to be attributed to the absence of closure axioms in the initial process description ($\sum_{pd_{\text{complex}}}$). This was the motivation behind the creation of the $\sum_{pd_{\text{closure}}}$ process description discussed in Section 4.3.4 of the report.

Throughout experiments 5.2.1-5.2.36 $T_{\text{lemma}_2}$ was consistently included while the background ontologies were varied. Experiments 5.2.1-5.2.12 included $T_{\text{actocc}} \cup \text{complex}$, experiments 5.2.13-5.2.24 included $T_{\text{complex}}$, and experiments 5.2.25-5.2.36 only included the process description and $T_{\text{lemma}_2}$. One must also note that the process description used in these experiments was $\sum_{pd_{\text{closure}}}$ as opposed to $\sum_{pd_{\text{complex}}}$ used in experiments 5.1.1-5.1.36. Similar to the findings mentioned above, Otter was unable to generate a proof to any of the queries as it ‘timed-out’ in the course of its reasoning. Here there is no identifiable improvement associated with the inclusion of $T_{\text{lemma}_2}$, across all ontology sizes.
Throughout experiments 5.3.1-5.3.45 $T_{\text{lemma}_3}$ was consistently included while the background ontologies were varied. Experiments 5.3.1-5.3.15 included $T_{\text{actocc}} \cup \text{complex}$, experiments 5.3.16-5.3.30 included $T_{\text{complex}}$, and experiments 5.3.31-5.3.45 simply included the process description ($\sum_{\text{pd\_closure}}$ ) and $T_{\text{lemma}_3}$. As outlined in Section 5.0 of the report, Otter was able to generate some favorable results, the proofs of which can be found in Appendix B. One can attribute these proofs to the inclusion of $T_{\text{lemma}_3}$, as it was the only variable modified as compared with experiments 5.2.1-5.2.36.

In order to determine how varying the ontology size affected the generation of proofs, the experiments that successfully returned proofs will be analyzed. Consider the following table:

**Table 7.1. Comparison of Experiments which generated proofs**

<table>
<thead>
<tr>
<th>Query</th>
<th>Experiment</th>
<th>Background Theory</th>
<th>Proof Generated?</th>
<th>Duration (s)*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$T_{\text{actocc}} \cup \text{complex}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a</td>
<td>5.3.1</td>
<td>✓</td>
<td>N</td>
<td>1831.37</td>
</tr>
<tr>
<td></td>
<td>5.3.16</td>
<td>✓</td>
<td>N</td>
<td>1800.17</td>
</tr>
<tr>
<td></td>
<td>5.3.31</td>
<td></td>
<td>Y</td>
<td>122.34</td>
</tr>
<tr>
<td></td>
<td>5.3.3</td>
<td>✓</td>
<td>Y</td>
<td>87.27</td>
</tr>
<tr>
<td></td>
<td>5.3.18</td>
<td>✓</td>
<td>Y</td>
<td>21.05</td>
</tr>
<tr>
<td>1c</td>
<td>5.3.33</td>
<td>✓</td>
<td>Y</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>5.3.4</td>
<td>✓</td>
<td>Y</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>5.3.19</td>
<td>✓</td>
<td>Y</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>5.3.34</td>
<td>✓</td>
<td>Y</td>
<td>0.01</td>
</tr>
<tr>
<td>2a</td>
<td>5.3.6</td>
<td>✓</td>
<td>Y</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>5.3.21</td>
<td>✓</td>
<td>Y</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>5.3.36</td>
<td>✓</td>
<td>Y</td>
<td>0.08</td>
</tr>
<tr>
<td>2c</td>
<td>5.3.8</td>
<td>✓</td>
<td>Y</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>5.3.23</td>
<td>✓</td>
<td>Y</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>5.3.38</td>
<td>✓</td>
<td>Y</td>
<td>0.01</td>
</tr>
<tr>
<td>3a</td>
<td>5.3.10</td>
<td>✓</td>
<td>Y</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>5.3.25</td>
<td>✓</td>
<td>Y</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>5.3.40</td>
<td>✓</td>
<td>Y</td>
<td>0.01</td>
</tr>
<tr>
<td>3c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Note: Duration here refers to the CPU time required to generate a result
Based on the results presented in this table a comparison of experiments resolving identical queries can be made. It is evident from these results that query 1a and query 1c benefited by reducing the size of the background theory. Although queries 2a, 2c, 3a, and 3c show some variation in duration by varying the ontology size, the fraction of a second improvements do not seem to be significant. Thus, it is evident that some of the reasoning experiments benefited by reducing the size of the background ontology, and including a set of axioms to reduce the computational requirements of the theorem prover.
8.0 CONCLUSION

By identifying the processes of the Hamilton Beach Water Dispenser and representing them using the PSL ontology, this thesis project successfully tested two hypotheses surrounding the improvement of automatically detecting inconsistencies in process specifications. The first hypothesis that was investigated surrounded the discovery and inclusion of lemmas to improve the performance of the theorem prover. The second hypothesis surrounded the reduction of the background ontology size used in the reasoning process. The results obtained from the experiments support both of the hypotheses investigated. As well, out of the experimentation a discovery of incompleteness in the PSL Ontology was uncovered. This thesis project was able to provide the research community with a corrected manual proof of the Transitivity axiom and the additional axiom required for this proof.

Some of the experiments performed were not able to generate a proof, although manual proofs do exist. Further work is needed to investigate why Otter failed in these experiments. To further the research done in this thesis project a trial of additional theorem provers can be investigated, as well the additional queries surrounding temporal projections and preconditions and effects introduced in this thesis project can be implemented. In addition to this, identifying additional axioms for use as lemmas can be useful in furthering this research.
9.0 REFERENCES


NIST. A Few PSL Basics…. 15 Feb. 2007. 6 Nov. 2007
<http://www.mel.nist.gov/psl/>


APPENDICES

Appendix A: Drain Process Modeling ................................................................. 111
Appendix B: Otter Proofs Generated Through Experiments ................................ 114
Appendix C: Otter Proof of Lemma 1 ................................................................. 119
Appendix A: *Drain* Process Modeling

Figure A1. – Process Description – *Drain* Process (Otter Syntax)

```prolog
all o (occurrence_of(o,drain(Water,Dispenser)) -> (exists o1 exists o2 exists o3 exists o4 exists o5 exists o6 exists o7 exists o8 exists o9 exists o10 (occurrence_of(o1,empty(Bottle)) & occurrence_of(o2,switchoff(Power,Dispenser)) & occurrence_of(o3,unplug(Dispenser)) & occurrence_of(o4,sit(Water,Dispenser)) & occurrence_of(o5,removebottle(Bottle,Dispenser)) & occurrence_of(o6,place(Dispenser,Sink)) & occurrence_of(o7,removeclips(Siliconstopper,Drainclip,Dispenser)) & occurrence_of(o8,drain2(Water,Dispenser)) & occurrence_of(o9,reinstall(Siliconstopper,Drainclip,Dispenser)) & occurrence_of(o10,turnon(Dispenser)) & min_precedes(o1,o2,drain(Water,Dispenser)) & min_precedes(o2,o3,drain(Water,Dispenser)) & min_precedes(o3,o4,drain(Water,Dispenser)) & min_precedes(o4,o5,drain(Water,Dispenser)) & min_precedes(o5,o6,drain(Water,Dispenser)) & min_precedes(o6,o7,drain(Water,Dispenser)) & min_precedes(o7,o8,drain(Water,Dispenser)) & min_precedes(o8,o9,drain(Water,Dispenser)) & min_precedes(o9,o10,drain(Water,Dispenser)) & subactivity_occurrence(o1,o) & subactivity_occurrence(o2,o) & subactivity_occurrence(o3,o) & subactivity_occurrence(o4,o) & subactivity_occurrence(o5,o) & subactivity_occurrence(o6,o) & subactivity_occurrence(o7,o) & subactivity_occurrence(o8,o) & subactivity_occurrence(o9,o) & subactivity_occurrence(o10,o)))).
```
Figure A2. – Subactivity Relationships – *Drain* Process (Otter Syntax)

```
subactivity(empty(Bottle),drain(Water,Dispenser)).
subactivity(switchoff(Power,Dispenser),drain(Water,Dispenser)).
subactivity(unplug(Dispenser),drain(Water,Dispenser)).
subactivity(sit(Water,Dispenser),drain(Water,Dispenser)).
subactivity(removebottle(Bottle,Dispenser),drain(Water,Dispenser)).
subactivity(place(Dispenser,Sink),drain(Water,Dispenser)).
subactivity(removeclips(Siliconstopper,Drainclip,Dispenser),drain(Water,Dispenser)).
subactivity(drain2(Water,Dispenser),drain(Water,Dispenser)).
subactivity(reinstall(Siliconstopper,Drainclip,Dispenser),drain(Water,Dispenser)).
subactivity(turnon(Dispenser),drain(Water,Dispenser)).
```

Figure A3. – Precondition Axioms – *Drain* Process (Otter Syntax)

```
all o (occurrence_of(o,empty(Bottle)) & legal(o) ->
  - prior(poweroff(Dispenser),o) ).
all o (occurrence_of(o,switchoff(Power,Dispenser)) & legal(o) ->
  prior(bottleempty(Bottle),o) ).
all o (occurrence_of(o,unplug(Dispenser)) & legal(o) ->
  prior(poweroff(Dispenser),o) ).
all o (occurrence_of(o,sit(Water,Dispenser)) & legal(o) ->
  prior(poweroff(Dispenser),o) ).
all o (occurrence_of(o,removebottle(Bottle,Dispenser)) & legal(o) ->
  prior(cooled(Water),o) ).
all o (occurrence_of(o,place(Dispenser,Sink)) & legal(o) ->
  prior(bottleoff(Bottle,Dispenser),o) ).
all o (occurrence_of(o,removeclips(Siliconstopper,Drainclip,Dispenser)) & legal(o) ->
  prior(placed(Dispenser,Sink),o) ).
all o (occurrence_of(o,drain2(Water,Dispenser)) & legal(o) ->
  -prior(installedclips(Siliconstopper,Drainclip,Dispenser),o) ).
all o (occurrence_of(o,reinstall(Siliconstopper,Drainclip,Dispenser)) & legal(o) ->
  prior(bottleempty(Bottle),o) ).
```
**Figure A4. – Effect Axioms – Drain Process (Otter Syntax)**

\[
\begin{align*}
\text{all } o \ ( \text{occurrence_of}(o, \text{empty}(\text{Bottle})) & \rightarrow \\
& \text{holds}(\text{bottleempty}(\text{Bottle}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{switchoff}(\text{Power, Dispenser})) & \rightarrow \\
& \text{holds}(\text{poweroff}(\text{Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{unplug}(\text{Dispenser})) & \rightarrow \\
& \text{holds}(\text{poweroff}(\text{Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{sit}(\text{Water, Dispenser})) & \rightarrow \\
& \text{holds}(\text{cooled}(\text{Water}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{removebottle}(\text{Bottle, Dispenser})) & \rightarrow \\
& \text{holds}(\text{bottleoff}(\text{Bottle, Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{place}(\text{Dispenser, Sink})) & \rightarrow \\
& \text{holds}(\text{placed}(\text{Dispenser, Sink}), o). \\
\text{all } o \ ( \text{occurrence_of} \\
& (o, \text{removeclips}(\text{Siliconstopper, Drainclip, Dispenser})) & \rightarrow \neg \\
& \text{holds}(\text{installedclips}(\text{Siliconstopper, Drainclip, Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{drain2}(\text{Water, Dispenser})) & \rightarrow \\
& \text{holds}(\text{tankempty}(\text{Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of} \\
& (o, \text{reinstall}(\text{Siliconstopper, Drainclip, Dispenser})) & \rightarrow \\
& \text{holds}(\text{installedclips}(\text{Siliconstopper, Drainclip, Dispenser}), o). \\
\text{all } o \ ( \text{occurrence_of}(o, \text{turnon}(\text{Dispenser})) & \rightarrow \neg \\
& \text{holds}(\text{poweroff}(\text{Dispenser}), o). \\
\end{align*}
\]

**Figure A5. – Process Description with Closure – Drain Process (Otter Syntax)**

\[
\begin{align*}
\text{all } o \ ( \text{occurrence_of}(o, \text{drain}(\text{Water, Dispenser})) & \rightarrow \\
& (\exists o1 \exists o2 \exists o3 \\
& (\text{occurrence_of}(o1, \text{empty}(\text{Bottle})) \\
& \& \text{occurrence_of}(o2, \text{unplug}(\text{Dispenser})) \\
& \& \text{occurrence_of}(o3, \text{turnon}(\text{Dispenser})) \\
& \& \text{min_precedes}(o1, o2, \text{drain}(\text{Water, Dispenser})) \\
& \& \text{min_precedes}(o2, o3, \text{drain}(\text{Water, Dispenser})) \\
& \& \text{subactivity_occurrence}(o1, o) \\
& \& \text{subactivity_occurrence}(o2, o) \\
& \& \text{subactivity_occurrence}(o3, o) \\
& \& \text{all } s \\
& (\text{subactivity_occurrence}(s, o) \\
& \leftrightarrow \neg \neg (s=\text{null} \or s=\text{null} \or s=\text{null}))). \\
\end{align*}
\]
Appendix B: Otter Proofs Generated Through Experiments

Figure B1. – Proof of Experiment 5.3.3

---------------- PROOF ----------------

134 [] -min_precedes(x,y,z)| -min_precedes(y,u,z)|min_precedes(x,u,z).
149 [] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f40(x),unplug(Dispenser)).
150 [] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f39(x),turnon(Dispenser)).
151 [] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f41(x),$f40(x),drain(Water,Dispenser)).
152 [] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f40(x),$f39(x),drain(Water,Dispenser)).
160 [] -activity(x)|occurrence_of($f42(x),x).
161 [] -activity(x)| -occurrence_of(y,unplug(Dispenser))| -occurrence_of(z,turnon(Dispenser))| -min_precedes(u,y,x)| -min_precedes(u,z,x)| -min_precedes(y,z,x)| -$ans(x).
199 [] activity(drain(Water,Dispenser)).
418 [hyper,199,160] occurrence_of($f42(drain(Water,Dispenser)),drain(Water,Dispenser)).
1678 [hyper,418,152]
min_precedes($f40($f42(drain(Water,Dispenser))),$f41($f42(drain(Water,Dispenser))),drain(Water,Dispenser)).
1679 [hyper,418,151]
min_precedes($f41($f42(drain(Water,Dispenser))),$f40($f42(drain(Water,Dispenser))),drain(Water,Dispenser)).
1680 [hyper,418,150] occurrence_of($f39($f42(drain(Water,Dispenser))),turnon(Dispenser)).
1681 [hyper,418,149] occurrence_of($f40($f42(drain(Water,Dispenser))),unplug(Dispenser)).
6005 [hyper,1679,134,1678]
min_precedes($f41($f42(drain(Water,Dispenser))),$f39($f42(drain(Water,Dispenser))),drain(Water,Dispenser)).
6116 [hyper,6005,161,199,1681,1680,1679,1678] -$ans(drain(Water,Dispenser)).

------------ end of proof ------------

Figure B2. – Proof of Experiment 5.3.4

---------------- PROOF ----------------

165 [] -activity(x) $ans(x).
197 [] activity(sanitize(Dispenser)).
198 [binary,197.1,165.1] $ans(sanitize(Dispenser)).

------------ end of proof ------------

Figure B3. – Proof of Experiment 5.3.6

---------------- PROOF ----------------

165 [] -activity(x) $ans(x).
197 [] activity(sanitize(Dispenser)).
198 [binary,197.1,165.1] $ans(sanitize(Dispenser)).

------------ end of proof ------------
Figure B4. – Proof of Experiment 5.3.8

--------------- PROOF ----------------
165 [] -activity(x)|$ans(x).
197 [] activity(sanitize(Dispenser)).
198 [binary,197.1,165.1] $ans(sanitize(Dispenser)).

------------ end of proof -------------

Figure B5. – Proof of Experiment 5.3.10

--------------- PROOF ----------------
165 [] -activity(x)|$ans(x).
197 [] activity(sanitize(Dispenser)).
198 [binary,197.1,165.1] $ans(sanitize(Dispenser)).

------------ end of proof -------------

Figure B6. – Proof of Experiment 5.3.18

--------------- PROOF ----------------
69 [] -min_precedes(x,y,z)| -min_precedes(y,u,z)|min_precedes(x,u,z).
84 [] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f23(x),unplug(Dispenser)).
85 [] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f22(x),turnon(Dispenser)).
86 [] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f24(x),$f23(x),drain(Water,Dispenser)).
87 [] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f23(x),$f22(x),drain(Water,Dispenser)).
95 [] -activity(x)|occurrence_of($f25(x),x).
96 [] -activity(x)| -occurrence_of(y,unplug(Dispenser))| -occurrence_of(z,turnon(Dispenser))| -min_precedes(u,y,x)| -min_precedes(u,z,x)| -min_precedes(y,z,x)| -$ans(x).
110 [] activity(drain(Water,Dispenser)).
231 [hyper,110,95] occurrence_of($f25(drain(Water,Dispenser)),drain(Water,Dispenser)).
750 [hyper,231,87]
min_precedes($f23($f25(drain(Water,Dispenser))),$f22($f25(drain(Water,Dispenser))),drain(Water,Dispenser))
751 [hyper,231,86]
min_precedes($f24($f25(drain(Water,Dispenser))),$f23($f25(drain(Water,Dispenser))),drain(Water,Dispenser))
752 [hyper,231,85]
occurrence_of($f22($f25(drain(Water,Dispenser))),turnon(Dispenser)).
753 [hyper,231,84]
occurrence_of($f23($f25(drain(Water,Dispenser))),unplug(Dispenser)).
5674 [hyper,751,69,750]
min_precedes($f24($f25(drain(Water,Dispenser))),$f22($f25(drain(Water,Dispenser))),drain(Water,Dispenser))
5728 [hyper,5674,96,110,753,752,751,750] -$ans(drain(Water,Dispenser)).

------------ end of proof -------------
Figure B7. – Proof of Experiment 5.3.19

---------------- PROOF ----------------

100 \[
-\text{activity}(x)|\text{sans}(x).
\]
108 \[
\text{activity}(\text{sanitize}(\text{Dispenser})).
\]
109 \[
[\text{binary},108.1,100.1] \text{sans}(\text{sanitize}(\text{Dispenser})).
\]

------------ end of proof -------------

Figure B8. – Proof of Experiment 5.3.21

---------------- PROOF ----------------

100 \[
-\text{activity}(x)|\text{sans}(x).
\]
108 \[
\text{activity}(\text{sanitize}(\text{Dispenser})).
\]
109 \[
[\text{binary},108.1,100.1] \text{sans}(\text{sanitize}(\text{Dispenser})).
\]

------------ end of proof -------------

Figure B9. – Proof of Experiment 5.3.23

---------------- PROOF ----------------

100 \[
-\text{activity}(x)|\text{sans}(x).
\]
108 \[
\text{activity}(\text{sanitize}(\text{Dispenser})).
\]
109 \[
[\text{binary},108.1,100.1] \text{sans}(\text{sanitize}(\text{Dispenser})).
\]

------------ end of proof -------------

Figure B10. – Proof of Experiment 5.3.25

---------------- PROOF ----------------

100 \[
-\text{activity}(x)|\text{sans}(x).
\]
108 \[
\text{activity}(\text{sanitize}(\text{Dispenser})).
\]
109 \[
[\text{binary},108.1,100.1] \text{sans}(\text{sanitize}(\text{Dispenser})).
\]

------------ end of proof -------------
**Figure B11.** – Proof of Experiment 5.3.31

------------ PROOF ------------

1 \[] -root(x,y)|occurrence_of($f1(x,y),y).
3 \[] -min_precedes(x,y,z)|-min_precedes(y,u,z)|min_precedes(x,u,z).
18 \[] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f6(x),unplug(Dispenser)).
19 \[] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f5(x),turnon(Dispenser)).
20 \[] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f7(x),$f6(x),drain(Water,Dispenser)).
21 \[] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f6(x),$f5(x),drain(Water,Dispenser)).
29 \[] -activity(x)|root($f8(x),x).
30 \[] -activity(y)|-occurrence_of(y,unplug(Dispenser))| -occurrence_of(z,turnon(Dispenser))| -min_precedes(u,y,x)| -min_precedes(u,z,x)| -min_precedes(y,z,x)| -$ans(x).
38 \[] activity(drain(Water,Dispenser)).
145 [hyper,38,29] root($f8(drain(Water,Dispenser)),drain(Water,Dispenser)).
495 [hyper,145,1]
occurrence_of($f8(drain(Water,Dispenser)),drain(Water,Dispenser)),drain(Water,Dispenser)).
1668 [hyper,495,21]
min_precedes($f6($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))),$f5($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))).
1669 [hyper,495,20]
min_precedes($f7($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))),$f6($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))).
1670 [hyper,495,19]
occurrence_of($f5($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))),turnon(Dispenser)).
1671 [hyper,495,18]
occurrence_of($f6($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))),unplug(Dispenser)).
6240 [hyper,1669,3,1668]
min_precedes($f7($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))),$f5($f1($f8(drain(Water,Dispenser))),drain(Water,Dispenser))).
6268 [hyper,6240,30,38,1671,1670,1669,1668] -$ans(drain(Water,Dispenser)).

------------ end of proof ------------

**Figure B12.** – Proof of Experiment 5.3.33

------------ PROOF ------------

3 \[] -min_precedes(x,y,z)| -min_precedes(y,u,z)|min_precedes(x,u,z).
18 \[] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f6(x),unplug(Dispenser)).
19 \[] -occurrence_of(x,drain(Water,Dispenser))|occurrence_of($f5(x),turnon(Dispenser)).
20 \[] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f7(x),$f6(x),drain(Water,Dispenser)).
21 \[] -occurrence_of(x,drain(Water,Dispenser))|min_precedes($f6(x),$f5(x),drain(Water,Dispenser)).
29 \[] -activity(x)|occurrence_of($f8(x),x).
30 \[] -activity(x)| -occurrence_of(y,unplug(Dispenser))| -occurrence_of(z,turnon(Dispenser))| -min_precedes(u,y,x)| -min_precedes(u,z,x)| -min_precedes(y,z,x)| -$ans(x).
38 \[] activity(drain(Water,Dispenser)).
145 [hyper,38,29] occurrence_of($f8(drain(Water,Dispenser)),drain(Water,Dispenser)).
601 [hyper,145,21]
min_precedes($f6($f8(drain(Water,Dispenser))),$f5($f8(drain(Water,Dispenser))),drain(Water,Dispenser))).
602 [hyper,145,20]
min_precedes($f7($f8(drain(Water,Dispenser))),$f6($f8(drain(Water,Dispenser))),drain(Water,Dispenser))).
603 [hyper,145,19] occurrence_of($f5($f8(drain(Water,Dispenser))),turnon(Dispenser))).
604 [hyper,145,18] occurrence_of($f6($f8(drain(Water,Dispenser))),unplug(Dispenser)).
min_precedes(\texttt{\$f7(\texttt{\$f8(drain\texttt{\(\langle\!\langle\texttt{Water,Dispenser}\rangle\rangle\})})\}),\texttt{\$f5(\texttt{\$f8(drain\texttt{\(\langle\!\langle\texttt{Water,Dispenser}\rangle\rangle\})})\}),\texttt{d\texttt{rain\texttt{\(\langle\!\langle\texttt{Water,Dispenser}\rangle\rangle\})}}).

---------------- PROOF ----------------

34 \[ -activity(x)\mid \text{ans}(x). \]
36 \[ \text{activity}(\text{sanitize}(\text{Dispenser})). \]
37 \[ \text{binary,36.1,34.1} \mid \text{ans}(\text{sanitize}(\text{Dispenser})). \]

------------ end of proof -------------

**Figure B13.** – Proof of Experiment 5.3.34

---------------- PROOF ----------------

34 \[ -activity(x)\mid \text{ans}(x). \]
36 \[ \text{activity}(\text{sanitize}(\text{Dispenser})). \]
37 \[ \text{binary,36.1,34.1} \mid \text{ans}(\text{sanitize}(\text{Dispenser})). \]

------------ end of proof -------------

**Figure B14.** – Proof of Experiment 5.3.36

---------------- PROOF ----------------

34 \[ -activity(x)\mid \text{ans}(x). \]
36 \[ \text{activity}(\text{sanitize}(\text{Dispenser})). \]
37 \[ \text{binary,36.1,34.1} \mid \text{ans}(\text{sanitize}(\text{Dispenser})). \]

------------ end of proof -------------

**Figure B15.** – Proof of Experiment 5.3.38

---------------- PROOF ----------------

34 \[ -activity(x)\mid \text{ans}(x). \]
36 \[ \text{activity}(\text{sanitize}(\text{Dispenser})). \]
37 \[ \text{binary,36.1,34.1} \mid \text{ans}(\text{sanitize}(\text{Dispenser})). \]

------------ end of proof -------------

**Figure B16.** – Proof of Experiment 5.3.40

---------------- PROOF ----------------

34 \[ -activity(x)\mid \text{ans}(x). \]
36 \[ \text{activity}(\text{sanitize}(\text{Dispenser})). \]
37 \[ \text{binary,36.1,34.1} \mid \text{ans}(\text{sanitize}(\text{Dispenser})). \]

------------ end of proof -------------
Appendix C: Otter Proof of Lemma 1

Figure C1. – Proof of $T_{\text{lemma}_1}$

---------------- PROOF ----------------
72 {} root(x,y)|occurrence_of($f19(y,x),y$).
73 {} root(x,y)|subactivity_occurrence(x,$f19(y,x)$).
132 {} occurrence_of(x,$c1$)| subactivity_occurrence($c2,x$).
163 {} root($c2,c1$).
164 [hyper,163,73] subactivity_occurrence($c2,f19(c1,c2)$).
165 [hyper,163,72] occurrence_of($f19(c1,c2),c1$).

------------ end of proof -------------