Efficient Reasoning about Time in a Domain-General Automatic Inference System: Insights from Cognitive Science Can Illuminate the Path to Human-Level Artificial Intelligence

By

Perrin Gregory Bignoli

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the degree of

MASTER OF SCIENCE

Major Subject: COGNITIVE SCIENCE

Approved:

_____

Professor Nicholas L. Cassimatis, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York
April 2012
(For Graduation May 2012)

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

There are a number of people who have been instrumental in helping me to complete the work that is described in this Masters Degree thesis. Of course, a huge host of individuals have influenced me throughout my life that must remain anonymous here for the sake of brevity. It is my sincere hope that they can draw some satisfaction from the fact that their friendship, time, effort, encouragement (and even antagonism) have all led, even in an indirect way, to what I would like to think is a contribution to science and the foundations of what may be a successful research career. I am so privileged and humbled to have known you all.

My odyssey to Troy and RPI began with the dedicated support and attention paid to me from my childhood until the present by my parents, Mr. Jerome H. Bignoli and Mrs. Kathleen S. Hill, my sister, Mrs. Callan M.R. Bittrich, my maternal grandparents, Mr. Gregory M. Leska, Sr. and Mrs. Ruth Leska, and my stepfather, Mr. Steven R. Hill, Sr. I am also indebted to the seemingly unbounded sequence of incredible teachers in the Johnson City Central School District and professors at Cornell University and RPI who have provided essential guidance in my education and personal intellectual development. Likewise, I owe much to the people who I came in contact with during my employment at the Air Force Research Laboratory in Rome, NY. Without them, I most likely would never have made the decision to enroll as a full-time PhD student at RPI. Dr. Paul Bello, who has since become a major figure at the Office of Naval Research, recognized my interest in Cognitive Science and Artificial Intelligence and introduced me to my advisor and his research group in RPI's Department of Cognitive Science. The past and present members of that research group, the Human-Level Intelligence Laboratory, have all been invaluable sources of inspiration and ideas. One could not hope for more dedicated colleges and collaborators. I must mention my specific gratitude to Dr. Arthi Murugesan, who directly contributed to the research addressed by the greater part of this thesis. Last, but tantamount to the inverse of least, is the enormity of the influence on me by my advisor, Professor Nicholas Cassimatis. Let it be known that never has the role of a graduate advisor been filled by a more singularly dedicated, supportive, courageous, and inspiring human being.

ABSTRACT

The fields of Cognitive Science and Artificial Intelligence can gain much from incorporating the results of each field into the other. Although many researchers are aware of the close link between the two disciplines, fewer are aware of how tightly integrated the work in them is possible to be. A dedicated research program of uncovering the underlying substrate of the human cognitive system in tandem with implementing functional equivalents them in conjunction with powerful computational algorithms has been shown to be highly successful in making progress towards synthetic general human-level intelligence.

This work focuses on extending an efficient, domain-general approach, satisfiability solving, which has long been used in artificial reasoning systems, with specialized mechanisms for representing and conducting inference about time and change. Despite being suited for a number of inference, planning, and constraint-satisfaction tasks, previous methods for satisfiability solving are unable to gracefully scale up to real-world size instances of those types of problems. To address this issue, it is necessary to introduce a formal system for representing a logic of temporal interval objects in a way that enables tractable problem instances to be described. Through the identification of algorithms for efficient temporal inference via constraint graphs, rule matching, and content-addressable memory, it can be shown that this formulation allows a significant extension to the domains in which satisfiability solvers are effective.

Using a formulation of the Polyscheme Cognitive Architecture that is capable of utilizing a DPLL-like satisfiability solving algorithm to perform this integration has enabled the creation of a new system that outperforms state-of-the-art Satisfiability solvers, namely LazySAT and MiniMaxSAT. Moreover, this novel approach is demonstrated to facilitate reasoning about the common sense change minimization problem, a sub-problem of the frame problem, which has long been known to be a barrier to the engineering of intelligent agents that must interact with a dynamic environment.

1. Introduction and Literature Review

1.1 Overview

The fields of Cognitive Science (CogSci) and Artificial Intelligence (AI) share a common goal of understanding systems that exhibit intelligence. Although the specific methodologies and results of these fields may seem incompatible to a first approximation, it can be shown that both fields actually have a great potential to contribute to the other without resorting to coerce their research programs into a strained compatibility. Namely, the findings of CogSci pertaining to the evolutionary history and resultant operating constraints of the human cognitive system can be used directly to produce algorithms in AI that are more likely to solve human-level problems. These algorithms in turn help to generate unifying theories of the operation of the human cognitive system by formalizing how apparently disparate cognitive phenomena can be understood through the operation of a small set of computational principles. One extremely promising avenue has been to combine computational mechanisms derived from the insight that the human cognitive system evolved to reason about the physical world with the discovery of efficient algorithms for domain-general inference.

This work illustrates the potential of this approach by outlining how employing computational methods that emulate the way that humans conceptualize time can enable automatic reasoning systems to efficiently solve classes of problems that have proved to be intractable in practice. In order to engineer such a system, findings obtained through CogSci research were used to refine an algorithm for solving the Satisfiability (SAT) problem [1] that is widely used for general-purpose reasoning in AI research.

The technique of mapping problem instances in domains of interest to domain-general SAT instances has been used effectively in many inference, planning, and constraint satisfaction tasks. This discovery is considered to be a significant advance to towards human-level AI. However, it can be shown that many problems involving explicit reasoning about temporal intervals, which are easily solved by humans, pose great difficulties for existing SAT solvers even for relatively small problem instances.

Portions of this chapter previously appeared as: P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in AGI-10, Lugano, Switzerland, 2010.

Although there are a number of factors, both scientific and sociological, that have contributed to the failure to achieve human-level intelligence performance from modern technology, two properties common to most current AI systems have identified as major contributors to this state of affairs. The first is the preference of purely statistical systems over those that can also employ knowledge-based representations. The second is the inability of existing AI systems to adapt to new knowledge that is generated during the reasoning process itself. Both of these issues are addressed here by introducing a mechanism of the Polyscheme Cognitive Architecture (Polyscheme) that enables the positing of new temporal intervals during the course of inference that are licensed by the actions of the system itself [2].

Evaluations of this extension to Polyscheme against state of the art SAT-solvers are conducted to demonstrate the increase in computational efficiency gained on the path planning problem in the physical reasoning domain. Additionally, an implementation of content addressable memory in Polyscheme system is shown to enable it to solve the commonsense reasoning problem of change minimization on larger problem instances.

1.2 Introduction and Literature Review

That the fields of CogSci and AI are highly related and have the potential to greatly benefit from a research program that combines them in a principled way [3]. On one hand, many AI researchers are frustrated by the fact that cognitive scientists have a predilection to "play 20 questions" with nature [4], to rely too heavily on laboratory-based tasks with little validity [5], or to focus too much on modeling the fine-grained psychometrics of human cognition over investigating richer accounts of mental phenomena [6]. On the other hand, many cognitive scientists, as well as philosophers of cognition, doubt whether it is possible for machines to think. Two observations, which may appear unrelated on the surface, can be made concerning results obtained from research that has been conducted in the fields of CogSci and AI. The first observation is humans have the capacity to solve problems that are heavily dependent on time. The second observation is that there exist domain-general algorithms for solving broad classes of computational problems. Because it can even be suggested that the above statements would be seen as unrelated by many CS and AI scientists is proof enough that

there exists an intolerable level of disconnect between these fields. As a case study, a surface scan of vision-recognition literature in AI demonstrates that a great deal work in that area is conducted almost entirely independently of results in CogSci, despite the amount of evidence that the human visual perception system is directly influenced by other cognitive knowledge in processing ambiguous stimuli [7].

Much evidence can be cited in support of the first observation. Common sense alone indicates that temporally complex environments are the typical arenas in which the human cognitive system operates. Problems that arise in everyday experience, such as reasoning about the physics of objects [8], event comprehension [9], and natural language processing [10], clearly involve comprehending the logical structure of time in some way. Not only is time essential to performing fundamental human activities, but also studies have generated objective models that suggest that humans actually have dedicated systems for reckoning of time within their broader cognitive framework [11]. Clearly, humans have some ability to modify their behavior based on the perception of time, but an even stronger claim, that the human nervous system is predicated on their being a degree of dynamism in the information that it processes, has been made in theories of cognitive architectures [12].

Many implications of the second observation are manifest in the field of AI and, more generally, in Computer Science. This can be seen in the work that led to the realization of the deep equivalence of problems that are now known as 'NP-complete [13].' One particular NP-complete problem, the satisfiability problem, is readily lendable to solving problems that can be described by predicate logic languages. Yet, the dynamics of temporally complex problem domains lead to one of the great mysteries of CogSci and AI: how some intelligent agents overcome the frame problem [14], which pertains to the fact that knowledge inferred about an ever-changing environment must be itself ever changing. Humans can solve complex problems that involve time; computers cannot. A research program that can analyze, understand, and ultimately bridge this gap would be a prime candidate for producing advancements towards the related goals of CogSci and AI.

Satisfiability testing methods have been used effectively in many inference, planning, and constraint satisfaction tasks and thus have been considered to be a

significant contribution towards the synthesis of artificial general intelligence. However, since SAT constraints are defined over atomic propositions, domains with state variables that change over time can lead to extremely large search spaces. This poses both memory-efficiency and time-efficiency problems for existing SAT algorithms.

Earlier work extended the Polyscheme Cognitive Architecture (Polyscheme) with mechanisms motivated solely to satisfy the need for reasoning about the physical word, which is hypothesized to be the evolutionary purpose of human cognition, to conduct inference in a way that was proven to be a computational homomorphism to SAT solving algorithms [15]. Further work integrated a mechanism with that system for introducing new objects during inference [16]. A third extension to Polyscheme is given here to demonstrate that by using a cognitively plausible representation of time, it is possible to provide an incremental specialization of the SAT algorithm towards the target domain of general human-level intelligence.

In this paper, these problems are addressed by introducing a formal predicate logic language that explicitly represents the temporal intervals over which the relations described by that language hold particular truth values. These temporal intervals are presented as a method for encoding time in a way that reduces the size of the search space required to solve problems encoded with that method relative to encodings that use a less sophisticated approach. However, it can be shown that such intervals cannot be used effectively by traditional SAT algorithms without significant modification to such algorithms. Namely, a method for satisfying constraints expressed in the new language is given that enables automated systems for domain-general reasoning to efficiently solve problem formulations that encode such temporal intervals. Evaluations of this system against other state of the art SAT solvers are presented to quantify the benefits of adopting this approach in AI research.

Many AI applications have been successfully framed as SAT problems: planning [17], computer-aided design [18], diagnosis [19], and scheduling [20]. Although SAT-solvers have successfully handled problems with millions of clauses, tasks that require an explicit representation of time can exceed their capacities.

Adding a temporal dimension to a problem space has the potential to greatly expand search space sizes because SAT algorithms propositionalize relational

constraints. The most direct way to incorporate time is to have a copy of each state variable for every time point over which the system reasons. This increases the number of propositions by a factor equal to the number of time points involved. Since SAT algorithms generally become slower as the size of a problem increases, adding a temporal dimension to even relatively simple problems can make them intractable.

Although problems with time require more space to encode, the true expense of introducing time stems from the additional cost required to find a SAT solution. Consider a task that requires the comparison of all the possible ways that a car can visit three locations in order. If the problem has no reference to time points, there is only one solution: the car just moves from *a* to *b* to *c*. On the other hand, there are clearly more possibilities, since the car could potentially move or not move at every time. Compared to other SAT-solvers, LazySAT [21] which lazily instantiates constraints, is less affected by the increased memory demands of larger search spaces. Unfortunately, lazy instantiation will not increase the tractability of larger problems with respect to runtime.

There is, however, a more efficient way of representing time. Since it is unlikely that the truth value of a proposition will change at every time point, temporal intervals can be used to denote segments of contiguous time points over which its value is constant. This practice alleviates the need to duplicate all propositions at every time point for most problem instances, thus significantly reducing the search space size. Intervals also mitigate the arbitrary granularity of time because they are continuous and scale independent.

However, existing SAT solvers cannot process intervals efficiently because they do not allow new objects to be introduced during the course of inference. It is clearly impossible to know exactly which temporal intervals will be required. Therefore, every possible interval must be defined in advance. If *n* time points that are required to encode a given problem instance, then there are $\frac{(n-1)(n-2)}{2}$ unique time intervals that can be defined over those time points. For even a relatively small *n*, this number is large enough for there to be little advantage to use temporal intervals under these conditions with current searching methods.

To capture the benefits of SAT while supporting the use of intervals, an integrated system was created that combines a DPLL-like search with several specialized forms of

inference: a rule matcher, constraint graphs, and memory retrieval. Rule matching allows the Polyscheme system to both lazily instantiate constraints and introduce new objects during inference. Constraint graphs compactly store temporal and identity relationships between objects. Memory retrieval supports common sense reasoning about time. Although SAT has previously been applied to planning [22] and reasoning [23] with the event calculus, the presented approach is novel.

2. A Logical Theory of Temporal Intervals[1]

A logic language is specified below that permits the expression of relationships between objects at different points in time. This language incorporates temporal intervals and a mechanism for introducing new objects during inference. For example, in path planning tasks, it is often useful to have a condition such as: If at location *p1* and must be at location *p2*, which is not adjacent to *p1*, then move to some location *px* that is adjacent to *p1* at the next possible time. This constraint would be written as:

$$\begin{array}{c} Location(?x, ?p1, ?t1) \wedge Location(?x, ?p2, ?t2) \wedge \neg Same(?p1, ?p2, E) \\ \rightarrow (\infty) \\ Adjacent(?p1, ?px, E) \wedge Meets(?t1, ?t2, E). \end{array} \quad (1)$$

Note that arguments with the prefix '?' denote variables. Variables permit this one constraint to apply to all locations in the system. Additionally, weights can be assigned to constraints as a measure of their importance. Because this constraint holds for any path, it is given an infinite weight to indicate that it must be satisfied. Formally, constraints have the form:

$$A\_1 \wedge ... \wedge A\_m \rightarrow (w)\ C\_1 \wedge ... C\_n, \quad (2)$$

where w is a positive rational number, $m \geq 0$ and $n \geq 1$. *Ai* and *Cj* are first order literals. Literals have the form:

$$P(arg1, ..., argn), \quad (3)$$

where *P* is a predicate, *argi* is a term, and *argn* must be a time point or interval.

Terms that are prefixed with '?' are variables; others are constant "objects." A grounded predicate is one in which no term is a variable. Predicates specify relations over the first n-1 terms, which hold at time *argn*. There is a special type of relation called an attribute. Attributes are predicates, *P*, with three terms, *o*, *v*, and *t*, such that only one relation of the form *P(o, v, t)* holds for each *o* at every *t*. The negation of a literal is expressed with the ¬ operator. Every literal is mapped to a truth value.

Portions of this chapter previously appeared as: P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in AGI-10, Lugano, Switzerland, 2010.

If all of the literals in a constraint are grounded, then the constraint itself is grounded. Only grounded constraints can be satisfied or broken, according to the truth value of its component literals. A constraint is broken iff every antecedent literal is assigned to true and every consequent literal is assigned to false. The cost of breaking a constraint is given by w, which is infinite if the constraint is hard.

Some predicates and objects are included in the language. For instance, Meets, Before, and Includes are temporal relations that are similar to the predicates in the Allen interval calculus [24]. A set of objects is reserved, which have the forms {*t1*, …, *tn}*, where n is the number of times in the system. This set is known as the native times. Another time, E, denotes eternity and is used in literals whose assignments do not change over time.

A model of a theory in this language consists of a complete assignment, which is a mapping of every literal to a truth value. Valid models are those such that their assignment permits all hard constraints to be satisfied. All models have an associated cost equal to the cost of its broken constraints. Each theory has many valid models, but it is often useful to find one of the models with the minimum cost. For instance, this process can perform change minimization, a form of commonsense reasoning motivated by the frame problem [25].

3. An Implementation for a Temporal Interval Reasoner

3.1 System Architecture

An integrated system was created using the Polyscheme cognitive architecture [26] in order to efficiently solve problems with time. This approach allowed us to glean the benefits of SAT while capitalizing on the properties of specialized forms of inference. It is easiest to describe how the Polyscheme system works by framing it as a DPLL-like search. DPLL [27] performs a branch-and-bound depth first search that is guaranteed to be complete for finite search spaces. The algorithm searches for the best assignment by making assumptions about the literals that appear in its constraint set. An assumption consists of selecting an unassigned literal, setting it to true or false, and then performing inference based on this assignment. When necessary, the search backtracks to previous decision points to explore the ramifications of making the opposite assumption.

The DPLL-FIRST procedure in Algorithm 1 takes a set of constraints, *c*, as input and outputs an assignment of literals to truth values that minimizes the cost of broken constraints. In the input constraint set, there must be at least one fully grounded constraint with a single literal. Such constraints are called facts. Within the DPLL-FIRST procedure, several data structures are declared and passed to DPLL-RECUR, which is illustrated in Algorithm 2. First, there is a structure, *assign*, which stores the current assignment of literals to truth values. The facts specified in the input are stored in *assign* with an assignment that corresponds to the valence of the fact's literal. Second, there is a queue, *q*, which stores the literals that have been deemed relevant by the system in order to perform inference after the previous assumption. At the beginning, *q* contains the facts in the input. Third, *b* stores the best total assignment that has been found so far. Fourth, the cost of *b* is stored in *o*. Initially, *b* is empty and *o* is infinite.

Algorithm 1: DPLL-FIRST(*c*)

return *DPLL-RECUR(c, assign, q, o, b)*

Portions of this chapter previously appeared as: P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in AGI-10, Lugano, Switzerland, 2010.

Each time DPLL_RECUR is called, it performs an elaboration step that infers new assignments based on the current assumption. Initially, when there is no assumption, the elaboration step attempts to infer new information from the constraints specified in the input. After elaboration, the current assignment is examined to determine if one of the three termination conditions is met. The first condition is if the assignment is contradictory because the same literal has been assigned to both true and false. The second condition is if the cost of the current assignment exceeds the lowest cost of a complete assignment that has been discovered in previous iterations. Since new assignments can never reduce the total cost, it is unnecessary to continue searching. The third condition is if the current assignment is complete. In all of these cases, the search backtracks to a previous assumption and investigates any remaining unexplored possibilities. Afterwards, the search selects an unassigned literal and creates two new branches in the search tree: one where the literal is assumed to be true and one where it is assumed to be false. DPLL-RECUR is then invoked on those subtrees and the assignment from the branch with the lower cost is returned.

Algorithm 2: DPLL-RECUR(*c*)

```
call ELABORATION(c, assign, q)

if Contradictory(assign) then
        return Fail
else if Cost(assign) > o
        return Fail
else if Complete(assign)
        return assign
end if

u ← next element of q

new_assign ← assign with u assigned to true
b1 ← call DPLL-RECUR(c, new_assign, o, b)
new_assign ← assign with u assigned to false
b2 ← call DPLL-RECUR(c, new_assign, o, b)

if Cost(b1) < Cost(b2) then
        return b1
else
        return b2
end if
```

The elaboration step in basic DPLL is called unit-propagation. Unit-propagation examines the current assignment to determine if there are any constraints that have exactly one literal unassigned. If such constraints exist and exactly one assignment (i.e., true or false) for that literal satisfies the constraint, then DPLL makes that assignment immediately instead of through a later assumption. The Polyscheme system augments this basic technique by introducing several more specialized forms of inference.

To understand the importance of elaboration, consider that all of the best available complete SAT-solvers are based on some version of DPLL [28], [29]. DPLL is so effective because its elaboration step eliminates the need to explore large numbers of unnecessary assumptions. It is more efficient to infer assignments directly rather than to make assumptions, because each assumption is equivalent to creating a new branch in DPLL's abstract search tree. Elaboration also allows early detection of contradictions in the current assignment.

Despite its elaboration step, DPLL is unable to handle the large search spaces that occur when time is explicitly represented. The goal of the Polyscheme approach is to improve elaboration by using a combination of specialized inference routines. Previous work [30] has outlined the implementation of SAT solvers in Polyscheme. Following that approach, the DPLL algorithm was implemented using Polyscheme's focus of attention. One call to DPLL-RECUR is implemented by one focus of attention in Polyscheme. Logical worlds are used to manage DPLL assumptions. For each assumption, an alternative world is created in which the literal in question is either true or false. Once Polyscheme focuses on an assumption literal, it is elaborated by polling the opinions of several specialists. These specialists implement the specialized inference routines upon which the Polyscheme system relies. One of these specialists, the rule matcher, lazily instantiates grounded constraints that involve the current assumption. The assignment of a literal is given by Polyscheme's final consensus on the corresponding proposition. This elaboration constitutes the main difference between the Polyscheme system and standard DPLL.

The Polyscheme system's elaboration step, which is illustrated in Algorithm 3, loops over the literals that have been added to the queue because their assignments were modified by previous inference. Two procedures are performed on each of these literals.

First, a rule matcher is used to lazily instantiate grounded constraints from relevant variable constraints provided in the input. Relevant constraints are those that contain a term that corresponds to the current literal in focus. These constraints are "fired" to propagate truth value from the antecedent terms to the consequent terms. Newly grounded literals, which may contain new objects, are introduced during this process. Any such literals are added to the assignment store and the queue.

The second procedure involves formulating an assignment for the current proposition based on suggestions from the various components of the system. For instance, the temporal constraint graph is queried here in the case that the proposition being examined describes a temporal relationship. Likewise, the identity constraint graph would be queried if the examined proposition was an identity relationship. In the extended system, this is the step at which the memory retrieval mechanism would be utilized. These opinions are combined with the old assignment of the proposition to produce a new assignment. If the new assignment differs from the old one, the literal is placed back on the queue.

Algorithm 3: ELABORATION(*c, assign, q***)**

```
while q is not empty do
    l ← the next element in q
    ris ← call Match(l, c, q)
    delta ←∅
    for each ri in ris do
        delta ←delta ∩ call Propagate(ri, assign)
    end for
    rs ← the rule system's opinion on l
    tc ← the temporal constraint graph's opinion on l
    ic ← the identity constraint graph's opinion on l
    mr ← the memory retrieval system's opinion on l
    c ← call Combine(rs, tc, ic, mr)
    if c ≠ call ValueInAssignment(assign, l) then
        delta ← delta ∪ l
    end if
    q ← q ∪ delta
end while
return
```

3.2 Rule Matching Component

Lazy instantiation is efficient because it avoids the creation of constraints that do not need to be considered to produce an acceptable assignment. In this component, lazy instantiation is accomplished by treating constraints with open variables as templates that can be passed to a rule matcher. The rule matching component (RMC) attempts to bind the arguments of the last dequeued literal to variables in the constraint rules. A binding is valid if it allows all variables in the antecedent terms of a constraint to be bound to objects in the arguments of literals that are stored in the system's memory. All valid bindings are evaluated as constraints by propagating truth value from the antecedents to the consequents. A constraint is considered broken only if the grounded literals in its antecedent terms have been assigned to true and the propositions in its consequent terms have been assigned to false.

A simple example will illustrate the binding process. Let the literal currently being assigned be:

$$Location(car1, road, t1). \tag{5}$$

If the following constraint appears in the problem formulation:

$$Location(?x, ?p, ?t1 \rightarrow (10)\ Location(?x, ?p, ?t2) \wedge Meets(?t1, ?t2, E), \tag{6}$$

then the following fully grounded instance is created:

$$Location(car1, road, ?t1) \rightarrow (10)\ Location(car1, road, t\_new) \wedge Meets(t1, ?t\_new, E), \tag{7}$$

where *?x* binds to *car1*, *?y* binds to *road*, *?t1* binds to *t1*, and *?t2* binds to a new object, *t_new*. If *Location(car1, road, t1)* is assigned to true and *Location(car1, road, t_new)* is later assigned to false, then any models that contain that assignment will accrue a cost of 10.

3.3 Temporal Constrain Graph Component

As the number of objects in a problem instance increase, so do the number of literals and constraints. For instance, when time objects are introduced, it is often necessary to know how those times are ordered. If there are *n* times in the system,

approximately $\frac{n^2}{2}$ literals are required to represent all of the values of a binary relation over those times. Usually, only a small portion of these literals provide useful information for solving the problem. Instead of eagerly encoding all temporal relations, a component was created that could be queried on demand to determine if a given relation holds according to the current knowledge of the system. This component represents relations in graphical form.

Using a graph enables the system to derive new entailed relations without storing them explicitly as propositions. All of the fundamental temporal relationships described by Allen can be represented in the following way. Whenever a literal that involves such a relationship is encountered, the temporal constraint graph (TCGC) decomposes the time object arguments into three parts: the start point, the midpoints, and the end point. These parts form the nodes of the graph. Edges are created between two nodes in the graph if their temporal relationship is known.

Every interval relationship can be derived from only two types of relationships on the parts of times: Before and Equals. A time object's start point is defined to be before its midpoints and its midpoints are before its end point. By creating edges between the parts of different time objects, it is possible to record relationships between the objects themselves. For instance, to encode *Meets(t1, t2, E)* in the graph, one would use the relationship: *Equals(end-t1, start-t2, E).* To illustrate why the graph is an efficient way to store this information, consider the following example. If it is known that *Meets(t1, t2, E)* and *Meets(t2, t3, E)*, then the graph can be traversed to find *Before(t1, t3, E)*, among other relationships. Thus, these propositions are stored implicitly and do not need to be assigned unless they are present in a grounded constraint.

3.4 Identity Constraint Graph Component

The identity constraint graph component (ICGC) is similar to the TCGC, but it handles propositions about the identity of objects. This graph consists of nodes that represent objects and edges that represent either equality or inequality. By traversing the graph, it is possible to capture the transitivity of the identity relation. Although a rule could be used to generate the transitivity property, doing so has the potential to drastically increase the number of propositions over which DPLL must search.

Another important use for the ICGC is that it can detect inconsistencies in truth assignments to identity propositions. Consider that the following set of identity propositions is known: *Same(a, b)*, *Same(b, c)*. Then, the proposition, ¬*Same(a, c)*, is examined and assumed to be true. Clearly this is inconsistent, but without including a rule that defines the properties of identity, the system will continue to perform inference based on these facts until an explicit contradiction is encountered. Traversing the edges connecting *a*, *b*, and *c* in the graph will indicate that this scenario is contradictory. This information can be reported to the system as a whole during elaboration.

3.5 Memory Retrieval Component

Systems that explicitly represent time often also have to reason about change. However, in situations with imperfect knowledge, reasoning about change can be difficult. Consider the following information about the color of a car, assuming that *Location* is an attribute:

Color(car, red, t1), Color(car, blue, t6), Color(car, green, t11). (7)

In between *t1* and *t6* and *t6* and *t11*, it is consistent for the car to be any color. However, in many scenarios, the car would remain *red* until it was painted *blue* and then *blue* until it was painted *green*. Although the world is dynamic, it also exhibits inertia. The principle of change minimization states that changes to particular objects are relatively infrequent and when changes do occur, they will be salient. It is worthwhile for a reasoning system to bet on such recurring patterns, because doing so significantly reduces the complexity of many problems.

It is possible to frame the change minimization problem as a weighted SAT problem as follows. Given a set of attribute literals that involve the same first term, but whose second term changes over time, determine the minimum number of changes required to explain the data. To this end, constraints can be defined so that the least costly models will be those that exhibit the smallest possible number of attribute value changes. The memory retrieval component (MRC) is designed to accommodate this procedure by regulating which attribute values appear in grounded constraints. Only attribute values with some prior evidence in memory are considered. As an example, the

car could have been *purple* at *t2* and *brown* at *t3*, but models that contain such literals are automatically excluded.

To control which values are considered, the MRC makes a copy of each literal that encodes an attribute with a native time index. This copy is modified to contain a time index that corresponds to an open-ended interval. Only when new attribute values are observed will that interval's endpoints be constrained. For instance, when the system elaborates *Color(car, red, t1)*, a new literal *Color(car, red, t-car-red)* will be introduced. Because it uses a content-addressable retrieval system, the MRC will henceforth report that the color of the car is likely to be red at every time point until new information is discovered. If the literal *Color(car, blue, t6)* is observed, then the literal *Color(car, blue, t-car-blue)* will be introduced. Also, constraints will be added to limit the right side of the *t-car-red* interval at *t6* and the left side of the *t-car-blue* interval at *t1*.

Even with this optimization, change minimization is expensive if performed naively, since there are $\frac{(n-1)(n)}{2}$ possible transitions between *n* attribute values if nothing is known about when the values hold relative to each other. For instance, if a car is seen to be *red*, then *blue*, then *green*, the naïve formulation will consider such possibilities as a change from *green* to *red* even though this is clearly impossible. The change detection mechanism in the MRC utilizes the TCGC in conjunction with content-accessible memory to significantly reduce the number of impossible changes that are considered by the system. When an attribute literal containing an interval is elaborated, the only values that are adjacent in time to the current interval will be considered. Since the intervals do not have completely fixed endpoints, these neighboring times can be detected by looking at what times the current interval could possibly include. For instance, once it has been established that there is a change between *t1* and *t6*, the interval created around *t1* can no longer include *t11*, so the location *t1* will not appear as a possible previous state of the location at *t11*.

4. System Evaluation Results and Discussion

4.1 Path-Planning

The Polyscheme system was designed to improve the efficiency of applying SAT-solving to problems that involve an explicit representation of time. In order to accomplish this goal, Polyscheme was used to implement a DPLL-like algorithm with specialized forms of inference that permitted the creation of new objects. The ability to introduce new objects allowed us to use intervals to reduce the search space of these problems.

In order to test the level of improvement gained by the ability to add new objects independently of other techniques, an evaluation was conducted in which the memory retrieval component deactivated. The selected task was optimal path planning through space. This space was represented in the form of a graph. Although the particular problem that was used did not model a changing environment, an explicit represent of time would be required if actions had side effects or if objects in the environment had changing states. For instance, a particular environment might contain walls that crumble over time. The system was employed to find the shortest valid path between two particular locations on the graph. Valid paths consisted of movements between adjacent locations that did not contain active obstacles. These types of problems are important to the object tracking and motion planning domains.

The Polyscheme system was first compared with LazySAT because it is similar to Polyscheme's approach in that it also supports the lazy instantiation of constraints. Through experimentation, the values of LazySAT's parameters enabled high performance on this task were determined. LazySAT, however, is based on a randomized local search, which is inefficient for many structured domains. Therefore, as a cross-validation measure, the same set of tests was executed on MiniMaxSAT [31], which is one of the best DPLL-based systematic SAT-solvers. Because Markov logic is not complete and cannot report unsatisfiability, the problem instances in the

---

Portions of this chapter previously appeared as: P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in AGI-10, Lugano, Switzerland, 2010.

experimentation were restricted to exclusively involve configurations of space in which a valid path existed.

The evaluation problem involved a 9-location graph with one obstacle, which had to be circumvented. This 9-location limit was adopted because the performance of all of the systems degraded on larger problems. To determine how well each system handled time, ten versions of the problem were created that with the range of 5 to 50 time points in increments of 5. For each condition and each system, 10 trials were conducted and recorded the average runtime. These results are displayed in Figure 4.1. While the Polyscheme system was consistently better than LazySAT, MiniMaxSAT outperformed both until it had to contend with more than 30 time points. The Polyscheme system required approximately constant time to solve the problem due to the fact that intervals make this task equivalent to the case of planning without explicit time points.

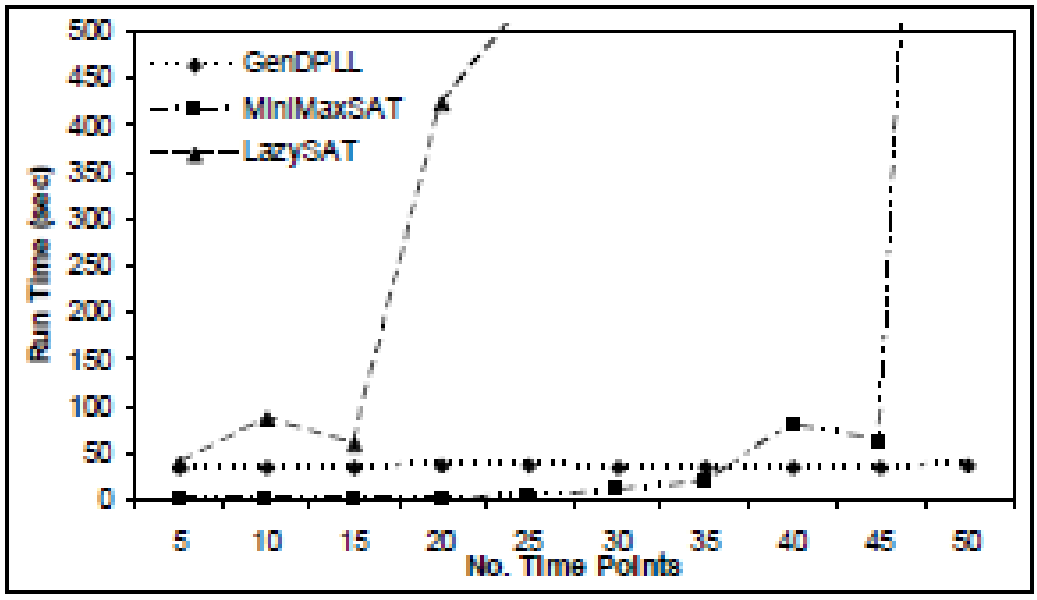

Figure 4.1. Path-Planning Results

Performance evaluation of Temporal Interval Polyscheme system (GenDPLL) with MiniMaxSAT and LazySAT on the Path-Planning Problem

4.2 Change Minimization

A second evaluation was conducted to show that the memory retrieval subsystem allowed change minimization problems to be solved efficiently. These tests were conducted on the Polyscheme system with and without the MRC activated. These

problems consisted of a number of attribute value observations that involved the color of an object. For instance, suppose the following facts were given as input: *Color(car, red, t1)*, *Color(car, blue, t6)*, and *Color(car, green, t11)*. The system would then be tasked with finding the least costly model that explained this data, namely that the color changed from *red* to *blue* and then from *blue* to *green*.

The results from this evaluation are depicted in Figure 4.2. Not only does enabling the MRC permit the change minimization problems to be solved in less time than with the basic system, but it also increases the upper limit on problem size. Without the MRC, the Polyscheme system ran for over 50,000 seconds attempting to solve the 5 attribute value problem. These results demonstrate that the elimination of irrelevant constraints is a powerful technique for improving the performance of SAT-solvers.

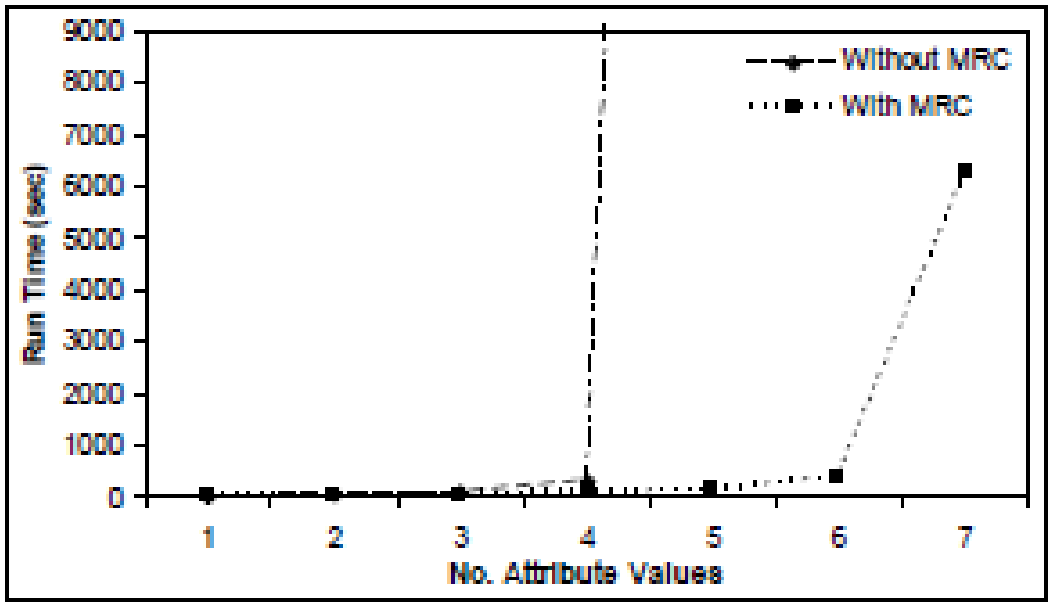

Figure 4.2. Change Minimization Results

Performance evaluation of Temporal Interval Polyscheme with the Memory Retrieval Component (MRC) enabled and disabled

5. Conclusion

5.1 Summary of Results

Although SAT is in some cases an efficient approach to domain-general problem solving, it does not scale well to the large search spaces that result from tasks that require an explicit representation of time. Temporal intervals help to reduce the size of such problems, but can be used effectively only with SAT-solvers that permit the introduction of novel objects throughout inference. Hence, a system was created that combines specialized inference techniques with a DPLL-like algorithm. This system was shown to outperform MiniMaxSAT and LazySAT in a series of evaluations involving a simple path planning domain.

When time is represented explicitly, it is also beneficial to incorporate common sense reasoning that exploits common patterns in real-world problem instances. Towards this end, a memory retrieval subsystem was developed that prevented the exploration of fruitless paths in the DPLL search tree under certain conditions. This technique was demonstrated to increase the efficiency of how the Polyscheme system solves the change minimization problem.

5.2 Related Work in Polyscheme

The research methodology described above has been repeatedly applied and refined to incrementally extend Polyscheme with new capabilities that bring it nearer to becoming a platform for the realization of synthetic human-level artificial intelligence. Although some of these modifications have required the introduction of new architecture-level theoretical constructs, the number of changes to the fundamental Polyscheme physical reasoning cognitive substrate theory has been remarkably small, especially in light of how much additional problem domain coverage these changes have produced. These advances have been made in the areas of spatial cognition [32], infant physical reasoning [33], counterfactual reasoning [34], and natural language processing [35].

Portions of this chapter previously appeared as: P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in AGI-10, Lugano, Switzerland, 2010.

Most of the evolution of the computational implementation of Polyscheme has been conducted at levels of abstraction well-below the theory level. For instance, there is no requirement in the overarching Polyscheme theory for a rule-matching system, let alone a specification for the particular rule matching algorithm that is employed. In the case of rule matching, the original rule matcher in Polyscheme used a naïve, brute force algorithm for generating all possible matches of rules to data, later versions made used of implementations of the RETE algorithm [36], and current configurations of Polyscheme use a graph-based inference procedure, similar in nature to the SNEPS system [37]. Polyscheme simply provides a framework for how such an algorithm can be integrated with other reasoning algorithms and methods of knowledge representation so that the combined strengths of those formalisms can be simultaneously brought to bear on the same problem instance. This is due to the fact that the engineering methodology employed, while committed to grounding itself in an actual implementation to constrain what mechanisms are proposed to those that are both cognitively and computationally plausible, has decoupled itself to a high degree from the actual theoretical level. Thus, the content of both the theoretical and implementation levels can be motivated both by findings in CS and AI, but in independent ways.

Viewed in the above light, Polyscheme is able to rapidly accommodate and adapt to new discoveries in both CS and AI without violating Newell's often-prescribed constraints on the responsible design of cognitive architectures. Perhaps the prime example of the flexibility of the Polyscheme approach is that, for years, the implementation did not incorporate any form of statistical inference, such as Maximum Entropy Modeling [38]. However, emerging directions in Polyscheme's development suggests that such methods can produce significant results on human-level intelligence tasks, such as Natural Language Processing [39]. This dynamism has been what has allowed Polyscheme to continue its success as a cognitive architecture over and with time.

6. References

[1] S. Russell and P. Norvig, "Constraint Statisfaction Problems," in *Artificial Intelligence A Modern Approach*, Upper Saddle River, NJ, Prentice Hall, 2009, pp. 202-230.

[2] P. Bignoli, N. L. Cassimatis and A. Murugesan, "Efficient Constraint-Satisfaction in Domains with Time," in *AGI-10*, Lugano, Switzerland, 2010.

[3] N. L. Cassimatis, "A Cognitive Substrate for Achieving Human Level Intelligence," *AI Magazine,* vol. 2, no. 27, pp. 45-56, 2006.

[4] A. Newell, "You Can't Play 20 Questions with Nature and Win," in *Visual Info. Process.*, W. G. Chase, Ed., New York, Academic Press, 1973, pp. 284-308.

[5] J. Carver, J. VanVoorhis and B. V., "Understanding the Impact of Assumptions on Experimental Validity," in *Proc. 3rd ACM-IEEE Int. Symp. Emp. Soft. Eng. (ISESE)*, 2004.

[6] J. Ball, A. Heiberg and R. Silber, "Toward a Large-Scale Model of Language Comprehension in ACT-R 6," in *Proc. 8th Int. Conf. on Cognitive Modeling*, Ann Arbor, MI, 2007.

[7] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature,* vol. 264, pp. 746-748, 1976.

[8] M. K. Teal and T. J. Ellis, "Spatial-Temporal Reasoning Based on Object Motion," *Proc. 7th British Mach. Vision Conf. (BMVC'96),* vol. 2, pp. 465-474, 1996.

[9] G. T. M. Altmann, N. C. Hindy and E. Kalenick, "Event comprehension and competition between multiple representations of the same object.," Ling. Depart. Stanford Univ., Sci. Rep., 2011.

[10] B. Comrie, Tense, Cambridge, MA: Cambridge University Press, 1985.

[11] H. van Rijn and N. Taatgen, "A model of parallel time estimation," in *Proc. 8th Int. Conf. on Cognitive Modeling*, Ann Arbor, MI, 2007.

[12] J. Hawkins and S. Blakeslee, On Intelligence, New York: Times Books, 2004.

[13] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the

Theory of NP-Completeness, New York: W. H. Freeman, 1979.

[14] J. McCarthy and H. P. J., "Some philosophical problems from the standpoint of artificial intelligence," *Mach. Intell.,* vol. 4, no. 463-502, 1969.

[15] Murugesan, A framework for representing and jointly reasoning over linguistic and non-linguistic knowledge, Ph.D. dissertation, Troy, NY: Rensselaer Polytechnic Inst., 2009.

[16] N. L. Cassimatis, A. Murugesan and P. Bignoli, "Inference with Relational Theories over Infinite Domains," in *Proc. 22nd Int. FLAIRS Conf.*, Sanibel Island, Florida, 2009.

[17] H. Kautz and B. Selman, "Unifying SAT-based and Graph-based Planning," *Int. Joint Conf. on Artificial Intell.,* vol. 16, no. 1, pp. 318-325, 1999.

[18] J. P. Marques-Silva and K. A. Sakallah, "Boolean Satisfiability in Electronic Design," *Design Automation Conf.,* no. 37, pp. 675-680, 2000.

[19] A. Smith and A. Veneris, "Fault Diagnosis and Logic Debugging Using Boolean Satisfiability," *IEEE Trans. Comp.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 10, pp. 1666-1621, 2005.

[20] R. Feldman and M. C. Golumbic, "Optimization Algorithms for Student Scheduling Via Constraint Satisfiability," in *Proc. Int. Joint Conf. on Artificial Intell.*, 1989.

[21] P. Singla and P. Domingos, "Memory-Efficient Inference in Relational Domains," in *Proc. 21st Nat. Conf. on Artificial Intell.*, Boston, 2006.

[22] M. Shanahan and M. Witkowski, "Event Calculus Planning Through Satisfiability," *J. of Logic and Computation,* vol. 14, no. 5, pp. 731-745, 2004.

[23] E. T. Mueller, "Event Calculus Reasoning through Satisfiability," *J. of Logic and Computation,* vol. 14, no. 5, pp. 703-730, 2004.

[24] J. Allen, "Maintaining Knowledge About Temporal Intervals," in *TR-86,* Comp. Sci. Depart., Univ. of Rochester, Rochester, NY, 1981.

[25] M. Shanahan, Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia, Boston, MA: M.I.T. Press, 1997.

[26] N. L. Cassimatis, "Polyscheme: A Cognitive Architecture for Integrating Multiple

Representation and Inference Schemes," Media Lab. Massachussetts Inst. of Tech., Cambridge, MA, 2002.

[27] M. Davis, G. Logemann and e. al., "A Machine Program for Theorem Proving," *Commun. of the ACM*, vol. 5, no. 7, pp. 394-397, 1962.

[28] M. Moskewicz and C. Madigan, "Chaff: Engineering an Efficient SAT Solver," in *Proc. 39th Design Automation Conf.*, Las Vegas, NV, 2001.

[29] N. Een and N. Sorensson, "MiniSat-A SAT Solver with Conflict-Clause Minimization," in *Proc. SAT 2005 Competition*, St. Andrews, UK, 2005.

[30] N. L. Cassimatis and M. Bugjaska, "An Architecture for Adaptive Algorithmic Hybrids," in *Proc. AAAI-07*, Vancouver, BC, 2007.

[31] F. Heras, J. Larrosa and e. al., "MiniMaxSAT: A New Weight Max-SAT Solver," in *Proc. Int. Conf. on Theory and Applicat. of Satisfiability Testing*, Lisbon, Portugal, 2007.

[32] U. Kurup, P. Bignoli, J. R. Scally and N. L. Cassimatis, "An architectural framework for complex cognition," *Cognitive Syst. Research*, vol. 12, no. 3-4, pp. 281-292, 2011.

[33] N. L. Cassimatis and P. Bignoli, "Microcosms for testing common sense reasoning abilities," *J. of Experimental & Theoretical Artifical Intell.*, vol. 23, no. 3, 2011.

[34] P. Bello, B. P and N. L. Cassimatis, "Attention and Association Explain the Emergence of Reasoning About False Belief in Young Children," in *Proc. 8th Int. Conf. on Cognitive Modeling*, Ann Arbor, MI, 2007.

[35] N. L. Cassimatis, A. Murugesan and M. D. Bugaksja, "A Cognitive Substrate for Natural Language Understanding," in *Artificial General Intell. 2008*, P. Wang and e. al., Eds., IOS Press, 2008, pp. 99-107.

[36] C. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artifical Intell.*, vol. 19, pp. 17-37, 1982.

[37] S. C. Shapiro, "An introduction to SNePS (semantic network processing system)," Comp. Sci. Dept., Indiana Univ., Bloomington, IN, Tech. Rep. 31, 1976.

[38] S. Della Pietra, V. Della Pietra and J. Lafferty, "Inducing features of random fields,"

IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, pp. 380-393, 1997.

[39] A. L. Berger, S. A. Della Pietra and V. J. Della Pietra, "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, vol. 22, 1996.