2012

# Faster conceptual blending predictors on relational time series

Tan, Terence K.; Darken, Christian J.

Proceedings of the 15th International Conference on Information Fusion 2012, IEEE, pp. 189-195.
http://hdl.handle.net/10945/36616

# Faster Conceptual Blending Predictors on Relational Time Series

**Terence K. Tan, Christian J. Darken**
The MOVES Institute
Naval Postgraduate School
700 Dyer Road, Watkins Ext. 265
Monterey, CA 93943-5001
ktan@nps.edu, cjdarken@nps.edu

***ABSTRACT- Tasks at upper levels of sensor fusion are usually concerned with situation or impact assessment, which might consist of predictions of future events. Very often, the identity and relations of target of interest have already been established, and can be represented as relational data. Hence, we can expect a stream of relational data arriving at our agent input as the situation updates. The prediction task can then be expressed as a function of this stream of relational data. Run-time learning to predict a stream of percepts in an unknown and possibly complex environment is a hard problem, and especially so when a serious attempt needs to be made even on the first few percepts. When the percepts are relational (logical atoms), the most common practical technologies require engineering by a human expert and so are not applicable. We briefly describe and compare several approaches which do not have this requirement on the initial hundred percepts of a benchmark domain. The most promising approach extends existing approaches by a partial matching algorithm inspired by theory of conceptual blending. This technique enables predictions in novel situations where the original approach fails, and significantly improves prediction performance overall. However an implementation, based on backtracking, may be too slow for many implementations. We provide an accelerated approximate algorithm based on best-first and A* search, which is much faster than the initial implementation.***

***Keywords-Pattern analysis, Machine learning, Reasoning under uncertainty, Relational Time Series, Learning, Prediction***

## I. INTRODUCTION

Sensor fusion plays an important role in translating the raw sensory output from a sensor to more usable data such as identity and relation of target object. The high level goals are usually for situation and impact assessment. The technologies that underpin these assessment capabilities usually assume the availability of domain knowledge. These technologies will be less useful if the domain is largely unknown, and call for new means to learn and make prediction. Learning from a percept sequence and predicting its future course is fundamental to agent cognition. A relational time series (RTS) is a percept sequence composed of logical ground atoms, as opposed to the commonly known time-series used to represent sampled floating point data at a constant time interval. Learning and prediction on RTS in unknown environments is a hard problem. The technologies available for this task are mainly based on production systems or statistical graphical model inferencing processes such as Bayesian networks. To apply these approaches, it is necessary that domain knowledge be known. Many time series approaches are either propositional and thus do not leverage the structural properties offered by a relational representation, or require prior knowledge of the environment coded into the predefined topological structure of a graphical model.

There are three contributions in this paper. The first is an argument for the situation learning approach [2] for learning relational time series in unknown environment. The second is a comparison of several prediction techniques in conjunction with the situation learning approach. The third is several computational approaches to single scope blending.

In this paper, we first define the problem of learning and prediction in RTS for unknown domains, and describe some challenging characteristics of the problem. We then briefly evaluate possible learning and prediction technologies for their suitability in RTS. Then we describe the situation learning approach [2], and report our works in exploiting this approach for prediction, particularly, techniques based on Variable Order Markov models (VOMM) (Begleiter 2004), Multiple Simple Bayesian (MSB), Simple Bayesian Mixture (SBM) and new approaches inspired by Single Scope Bending [3].

## II. RELATIONAL TIME SERIES

We define a relational time series (RTS) as a sequence of relational percepts. Each percept is a ground atom defined as $p_i = r(c_1, c_2, \ldots, c_m)$, where r is the predicate and $c_{j\in(1..m)}$ are constants that represent objects. An example of a RTS is given in Figure 1. There are two types of percept: point and interval. The point percept exists or is active for a point in time and immediately ceases to exist. For example, a percept that describes "a ball hitting the wall" becomes obsolete immediately after it occurred. An interval percept occurred and remains true until something happens that change its state. For example, a percept that describes "a ball is in the box" is true until the ball is removed. The interval percept has a '+' indicator in the predicate as shown in Figure 1. A percept that is true is said to be active. The interval percept becomes inactive when a special type of point percept arrives, indicated by '-' in the predicate.

| $P_i$ | RTS | Semantics |
|---|---|---|
| $P_1$ | (loc+ Ed road) | Ed is at location road |
| $P_2$ | (loc+ Fox1 road) | Fox1 is at location road |
| $P_3$ | (goE Fox1 east) | Fox1 is going east |
| $P_4$ | (loc- Fox1 road) | Fox1 is NOT at location road |
| $P_5$ | (loc+ Fox2 road) | Fox2 is at location road |
| $P_6$ | (goE Fox2 east) | Fox2 is going east |
| $P_7$ | (loc- Fox2 road) | Fox2 is NOT at location road |

Figure 1: An example of relational time series. '+' means that the current atom is currently active. '–' means that the previously active atom is removed. All atoms are point percept except for the '+' atom.

The prediction problem can then be defined as follows. Let $\{p_1 p_2 \ldots p_n\}$ be the sequence of percepts from the time the agent started learning till the present time, where i in $p_i$ refers to the running index of each incoming percept. A one-step prediction problem is then $\{p_1 p_2 \ldots p_n\} \vdash p_p$ where $\vdash$ is an operator that predicts that $p_p$ is the next most likely percept. A two-step prediction problem is defined as $\{p_1 p_2 \ldots p_n p_{p1}\} \vdash p_{p2}$, given that $\{p_1 p_2 \ldots p_n\} \vdash p_{p1}$. This means that the percept predicted by a one-step predictor is used for the second step prediction. The two-step prediction problem can be generalized to a multiple-step prediction problem.

Relational representation is a natural way to express the relations among the constants in both real and simulated worlds. We can use such a time series to learn the behavior of our opponents, or other agents. Furthermore, relational representation allows us to infer additional knowledge from the structural properties afforded by the relations among the constants. In particular, such structural properties can help to predict even atoms that we have not seen before.

Learning and prediction in RTS from unknown environments is a hard problem because of a set of challenging characteristics. (1) Since there is no knowledge of the environment, there can be no predefined statistical graphical model or structure for knowing what kinds of atom that will arrive next. This leads to the second characteristic, which is (2) arbitrarily many constants and relations of arbitrary arity. This results in a large state space. To make the matter worse, the sequence of percepts can be noisy, and a function of (3) a moving context, with different percept subsequences occurring in different contexts. These characteristics present many challenges and opportunities for research.

## III. Current Approaches for RTS

We discuss possible learning and prediction approaches for RTS by organizing them into Markovian or Non-Markovian approaches. Markovian approaches such as Markov Chains or Hidden Markov Models appear to be suitable for the task because the sequencing property of the time series is naturally captured by the Markov property. These approaches are also efficient for online learning, which can assimilate newly learned percept into the knowledgebase, refining its structure if necessary, and use it immediately. Structural agility is accomplished by the state transition matrix. Each state can transit probabilistically to different states. The main limitation of Markovian approaches lies in its limited generalization to novel situations due to strict ordering. For example, the current situation may be similar to one previously encountered but simply have the order of two percepts switched, or have extra percept (noise) in between two percepts, Markov models will not detect it and treat it as a new sequence. Furthermore, most Markov approaches treat relational atoms as propositions, and do not leverage the relational structure. Jaeger [5] describes an Observable Operator Model (OOM) that models a stochastic process to compute the probability distribution over all possible future sequences, given that a sequence of propositional observation has been observed. The learning process requires prior manual estimation of a dimension (a set of features), which is a potential limitation in an unknown environment. Even in known environments, choosing the set of appropriate features is difficult [7].

Non Markovian learning approaches do not regard or may relax the sequential order requirement of the RTS. These approaches usually have no or limited learning capability. Production system and finite state machine rely on expert knowledge. These approaches cannot encode rules with multiple possible consequences. Bayesian learning typically encapsulates expert knowledge in the form of causation structures that are usually fixed. The key assumption to these approaches is that domain knowledge is known, or examples are available for training before deployment. Structural or rule learning are usually limited and done offline due to their exponential complexity. Statistical Relational Learning (SRL) attempts to combine first order logic with statistical learning [4]. The relational learning addresses the relational structure while the statistical learning allows multiple possible consequences. SRL are usually modeled using graphical model such as Bayesian Network (BN) or Markov Network (MN), which require predetermination of some topological structures. Khosravi & Bina [6] highlighted a challenge from the complexity of inferencing because the size of the graph grows with the number of attributes and objects. Most inferencing methods are based on the standard Bayesian inferencing approaches. In the case of MN, inferencing approaches require the computation of the partition function, which make the inferencing process NP-complete. These methods are therefore not suitable for the RTS task.

The above methods either require domain knowledge, predefined inferencing structures or sensitive to noise. We propose a situational learning approach for learning a RTS that features structural agility in its learned knowledgebase, and allow multiple prediction techniques to be used.

## IV. A Situation Learning Approach to Learning and Prediction in RTS

Darken [2] introduces a novel and yet simple approach that process a RTS into a set of situations (not to be confused with the related notion of situation in situation calculus). The approach appears to use a sliding time window to identify sets of atoms called "situation". When a new atom arrives, this new atom forms a situation with older active atoms that are still active within the time window from the time stamp of the new atom. The next arriving percept becomes the predictive target atom of the situation just form. If this situation already exists in the knowledge base, the number of occurrence of this situation is incremented. Otherwise, this situation will be added into the

knowledge base. Instead of learning the entire RTS with one graphical model such as a BN or MN, the approach effectively generates multiple simple networks of two layers. This approach avoids the current challenges of structural learning in the statistical relational learning by turning the problem into a situation matching and simple inferencing process. This situation learning approach addresses all challenging characteristics of RTS in a natural manner.

### A. Situation Learning

Formally, the situation learning (SL) approach processes the percept sequence $\{p_1 p_2 \ldots p_n\}$ into smaller disjoint set of percepts (called a situation) $\{s_i\}, i = 1..S$ where S is an integer that defines the number of situation. Let $\tau_a(p_i)$ refers to the time in which the atom $p_i$ is active, $\tau_{max}(p_i)$ refers to the latest time in which $p_i$ is active, $\tau_c$ refers to the current time when a new atom $p_{new}$ is received and $\tau_w$ refers to the time window duration. $p_i \in s_c$ if $\tau_{max}(p_i) + \tau_w \geq \tau_c$ where $s_c$ is the current situation. Let $p_c$ refers to the percepts encountered after situation $s_c$. We write a consequence $c_c$ as a tuple, $c_c = (s_c, p_c)$, such that $p_c$ follows $s_c$.

Given a RTS as shown in Figure 1, the agent starts with zero knowledge and begins to learn the situations as soon as the first percept arrives as shown in Figure 2. A short time window slides through the RTS as the percepts arrive. If the percepts fall within the same window, or are still active in that window, they will be grouped together to form one "situation" as shown at the left column of the table. The percept that arrives next is the consequence of the situation, listed at the right column. The numbers describe the number of occurrences to facilitate statistical inferencing.

| | | | |
|---|---|---|---|
| {} | 1 | (loc+ Ed road) | 1 |
| {[loc+ Ed road]} | 2 | (loc+ Fox1 road) | 1 |
| | | (loc+ Fox2 road) | 1 |
| {[loc+ Ed road] [loc+ Fox1 road]} | 1 | (goE Fox1 east) | 1 |
| {[loc+ Ed road] [loc+ Fox1 road] [goE Fox1 east]} | 1 | (loc- Fox1 road) | 1 |
| {[loc+ Ed road] [loc+ Fox2 road]} | 1 | (goE Fox2 east) | 1 |
| {[loc+ Ed road] [loc+ Fox2 road] [goE Fox2 east]} | 1 | (loc- Fox2 road) | 1 |

Figure 2: A collection of situations (left column) and their associated prediction (right column)

The SL approach reduces a potentially huge RTS into a set of smaller situations. The process of prediction is thus reduced to choosing an appropriate situation for prediction inferencing. This approach allows the application of many inferencing techniques such as Variable Order Markov Models (VOMM), Multiple Simple Bayesian (MSB) networks, Simple Bayesian Mixtures (SBM).

### B. Prediction Techniques

Darken [2] provides two simple techniques of prediction, and offer some insights into other possible techniques of prediction. The two techniques are Statistical Look-up Table (SLT) and Variable Matching (VM). SLT searches the situation table to look for a situation that exactly matches the current situation. If a match is found, the percept that follows the matched situation with the highest number of occurrence will be the predicted percept. VM replaces all constants in the atom with variables. Multiple instances of a constant use the same variable. The matching of situations becomes the problem of variable matching with substitution. A substitution is a list of variable bindings, e.g. θ={?a/?b} where variable ?a from one situation is bound to variable ?b in another situation. SUBST(θ,α) denotes the result of applying substitution θ to situation α. A match is then defined as a bijection of variables between the current situation and a match situation in the table. Finding matches is a graph isomorphism problem. An example of the variable representation is shown in Figure 3.

| Constant | Variable |
|---|---|
| [loc+ Ed road] | [loc+ ?x ?y] |
| [loc+ Ed grass] | [loc+ ?x ?z] |

Figure 3: Constant versus Variables Representation

Darken [2] reports a prediction accuracy of about 45%, after 250,000 percepts have been processed. This performance is remarkable because there are more than 300 percepts to choose from. However, when we run the two techniques on 100 percepts only, simulating an unknown environment, the prediction accuracy drops to 10%. The mean number of unmatched cases is 80 and 78 for SLT and VM respectively. SLT suffers limitations such as strict perfect matching and not utilizing structural information offered by relational representation. VM also uses strict perfect matching, which is detrimental because most situations encountered are new.

SL supports prediction techniques such as Variable Order Markov Models (VOMM), Multiple Simple Bayesian (MSB) network, and Simple Bayesian Mixture (SBM). When the RTS is decomposed into a set of situations, we can build one simple Bayesian network for each situation with the predictive target atom as the parent nodes, effectively forming MSB networks. Since MSB cannot learn certain functions such as Exclusive-OR, we implemented SBM. SBM contains probability mixture densities, constructed by normalizing a linear combination of two or more Simple Bayesian Networks probability densities having the same domain and range. SBM is implemented using the Estimate & Maximize (EM) algorithm. VOMM is an extension to the Markov chain models in which a variable order is used in place of a fixed order. We implemented a VOMM model using context trees [1].

### C. Experiment

We compared the prediction performance in a benchmark environment that is used in [2] in which, an agent wanders around and performs actions randomly. Actions include "go eastward", "pick up weapon", "equipped weapon", "hit", and many more. There are other agents (monsters) in the environment such as goblins, trolls and dragons. There are three types of weapon: pitchfork, dagger and sword. Each weapon may be more effective against each type of monsters. Each time a monster is killed, it will leave behind a weapon.

Each monster, weapon, agent and location has a unique constant name. The sequence of percepts describes what the agent sees, such as its location, weapons, and other agents. Our prediction task is to predict the next percept, given the past percept sequence. The predicted percept $p = r(c_1, c_2, \dots, c_m)$ is said to be correct if the next new percept $p' = r'(c_1', c_2', \dots, c_m')$ is such that $p' = p, r' = r, c_i' = c_i \ where \ i = 1,2, \dots, m$ . If n is the number of correct prediction, prediction accuracy is $\frac{n}{t}$ where t is the total number of percepts.

Darken [2] tested the prediction performance by running the algorithms through more than 250,000 percepts. In this study, we want to know how the algorithms work in harsh and new environments. We clear off the memory after 100 percepts have been processed and examine the results after 40 batches of 100 percepts are processed. To simulate noisy environment, we randomly swapped the order of two atoms in the current situation. All experiments were run on a Dell XPS Laptop i7 1.87Ghz 16GB RAM with Windows 7.

### *D. Results*

The prediction accuracies are given in Figure 4. Each bar in the chart represents the mean prediction accuracy with its associated standard error of a predictor. From the standard error indicators, we can see that the differences are significant for at least alpha $\alpha = 0.05$ for a statistical student-T test with degree of freedom df=39. There is no significant difference between the SLT and VM, and both techniques are significantly worse off than the others. This is due to the strict requirement of exact matching. When the environment is unknown and noisy, the current situation can hardly match the learned situations in the memory. Figure 5 shows the mean number of no-match for 40 batches of 100 percepts. No-match occurs when the algorithm is unable to find a reasonable situation. VOMM, MSB and SBM have much lower number of No-Match.

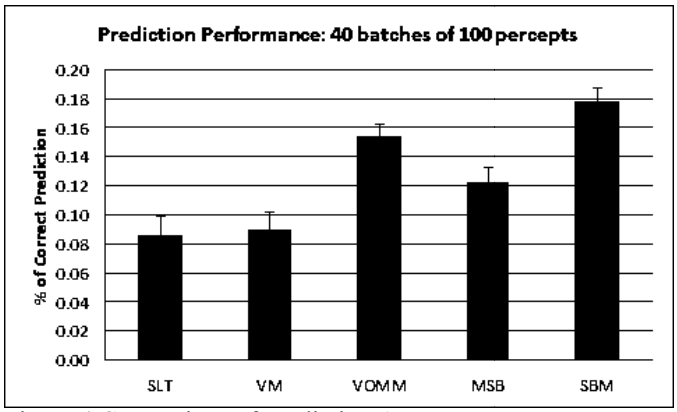

Figure 4 Comparison of Prediction Accuracy

The Markov Model is a popular approach in sequential and online learning, and handles novel situations better than SLT and VM. While the VOMM does not require exact atom to atom matching, and even allow partial matching, it requires exact sequential adjacency ordering. For example, the sequence of words [The Blue Fish is eating] will not match the sequence [The Fish is eating]. In addition, VOMM treats each atom as a proposition.

The multiple simple Bayesian network is able to handle novel situations with the Laplace method of assigning probabilities to newly encountered atoms. Its performance is low for several reasons. Firstly, there are too many novel percepts. The prior probabilities for each percept can be very low. The Laplace method assigns a probability that can be unfairly large to new atoms. Secondly, Bayesian network cannot handle exclusive-OR relation. There are atoms that are mutually exclusive. Thirdly, atoms in the sequence are not independent and identically distributed. The Bayesian mixture performs better than simple Bayesian classifier. However, it also suffers some of the problems found in Multiple Simple Bayesian.

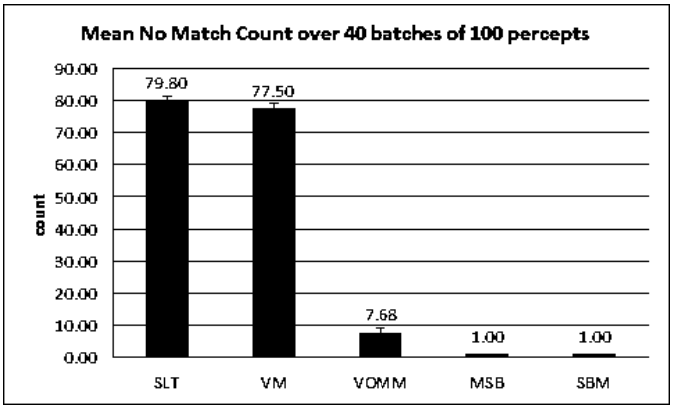

Figure 5 Comparisons of No-Match.

At this point, SBM appears to be the best performer. The challenging benchmark environment consists of too many novel situations for LUT, VM, VOMM, MSB and SBM. Any algorithm that attempts to excel in novel situation prediction may have to possess properties of human creativity. In the next section, we describe our exploration of the theory of Cognitive Integration, also known as Conceptual Blending [3].

## V. CONCEPTUAL BLENDING PREDICTORS

Conceptual blending (CB), a human cognition theory, was developed by Fauconnier and Turner [3] to explain how humans make sense of the world, through a process of imaginative blending of concepts to arrive at an understanding of a new environment. Each concept is equivalent to one situation in our task and contains a frame that describes the structural properties such as relations and constant types. Figure 16 describes the structure of two concepts given in Figure 15. There are four types of integrating network in CB, which are differentiated based on how frames from the two concepts in the blending network are used in the blend.

### *A. Single scope blending network*

We use the single scope blending (SSB) network, which describes the process of analogical inferencing (Figure 6). Figure 15 provides an example of the SSB network. In SSB network, both concepts have different frames since both situations have different set of relations, and constants of the same relation may be of different types. One of the frames is used in the blend. Here, Concept 1 corresponds to our learned situation (source) while Concept 2 refers to the current new situation (target). The source frame is projected onto the target situation.

*B. Cross space mappings*

Cross space mappings associate constants from both concepts. The mappings process can be expressed as a form of attempted graph bijection in which, each concept is viewed as a graph, with constants as nodes, and relations as links. We assume that atoms with arity more than 2 have been converted to an equivalent set of atoms with arity 2. Since two situations may be different, to overcome the high rate of unmatched instances in SLT and VM, we attempt to find the largest subgraph in one concept that can be matched to a subgraph in another concept. This is known as subgraph isomorphism, which is NP-Complete. We use a recursive backtracking method to identify the largest common subgraph [9] in both concepts.

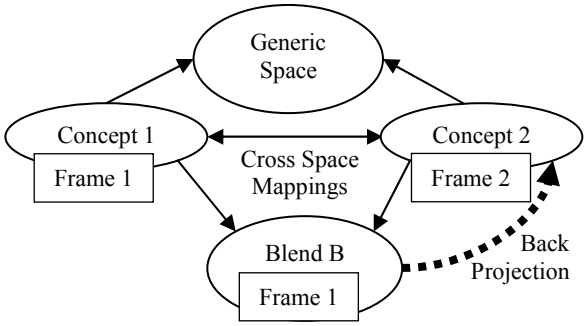


Figure 6: Single Scope Blending Network

Once the largest subgraph is identified, the constants in the subgraph of one concept can be substituted by the corresponding constants in the other subgraph since they are mapped in the subgraph isomorphism process. We call this method the substitution method. The substitution method is complete and optimal because it searches through all possible substitutions to find a set with maximum possible mappings. When every constant in one atom in concept 1 can be substituted by one constant in another atom in concept 2, we have one common atom. The total number of common atom is the similarity score.

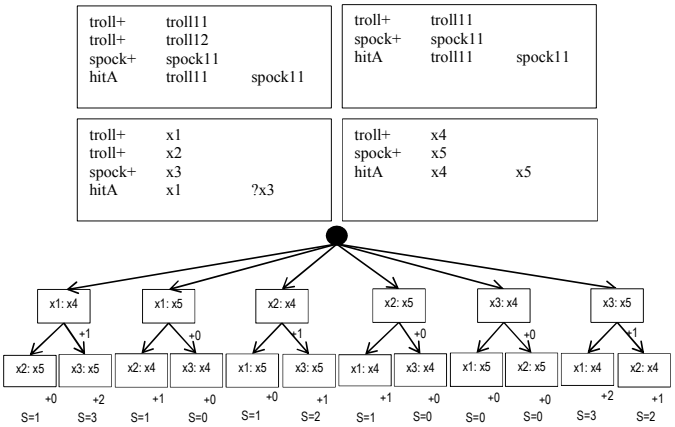


Figure 7 Backtracking partial matching process

Please refer to Figure 7 for an illustration of the backtracking implementation. The algorithm starts with a permutation of possible bindings. Each binding corresponds to a node in the tree in Figure 7. These nodes are pushed into the fringe, which is a stack that holds the yet-to-process nodes. Each node in the stack is popped, evaluated, and other possible bindings are then added into the stack. The evaluation is a simple counting of the number of common atoms in both situations if the binding is accepted. For example, if we accept the (x1:x4) binding at the first level, we have an additional score of one because the binding produces one common atom, which is troll+(x1) and troll+(x4). There are additional two possible bindings (x2:x5) and (x3:x5) consistent with this one. The (x2:x5) binding does not contribute any additional common atoms. Hence, the additional score is zero. Since there is no other unmapped nodes in concept 2 for x3, the algorithm backtracks to evaluate (x3:x5), which contributes two common atoms in conjunction with (x1:x4). A flow from the root to the leaf node constitutes one path. The total score for a path describes the similarity score of both situations based on the bindings in that path. The path [(x1:x4), (x2:x5)] has a score of one while the path [(x1:x4), (x3:x5)] has a score of three. The path with the highest possible score indicates the maximum possible similarity score between two situations. It is possible that more than one path has the same similarity score. We could choose the one that has the most number of types matched. For example, in the path [(x1:x4), (x3:x5)], x1 and x4 have the same type troll+ and x3 and x5 have the same type spock+. In the path [(x1:x4), (x2:x5)], x2 and x5 do not have the same type. Suppose that these two paths have the same similarity score, we will choose the first path, since it has two types matched while the later has only one. Hence, for paths that have the same similarity score, we could further sort them by number of types matched. (We call this TypeSort heuristic). In the example given in Figure 7, there are two paths that have the same similarity score and same number of types matched. In such situations, both paths will have the same set of nodes bindings. Experiments show that there is no difference whether we choose the one with the TypeSort heuristic, or choose arbitrarily.

The original backtracking approach has serious computational complexity problem. By looking at Figure 7, we see that there are redundancies in the search process. We introduce two heuristics to speed up the computation. The first heuristic is to terminate the search process, if we have found a path with the highest possible score (MaxLen). In the example in Figure 7, situation one has four atoms while situation 2 has three atoms. The maximum possible score is three. Hence, if we have found a path with a score three, we can safely terminate the search. Our experiments show that there is no difference in prediction accuracy for backtracking with and without the heuristic. However, the computation time improved by a factor of 80.

Another possible heuristic is to use the type checking to prune the search tree (TypePruning). If a possible binding consists of nodes of different type, that search branch is terminated. Figure 8 illustrates the nodes that are pruned by the heuristic. In the case that all nodes have different types, we will have a score of zero with no bindings. The search process will check only the first level of the tree. This heuristic affects the completeness of the search process because of the pruned branches.

The results from a simple experiment of 40 batches of 25 percepts, with fixed and random are shown in Figure 9 and Figure 10. The purpose of 40x25, instead of 40x100 percepts is because the no heuristic and TypeSort cases takes weeks to run an 40x100 experiment. Random refers to additional noise introduced to the current situation by randomly swapping the order of two atoms. Overall, the MaxLen and TypePruning heuristics reduce the computation time by more than 95% while maintaining similar levels of prediction accuracy.

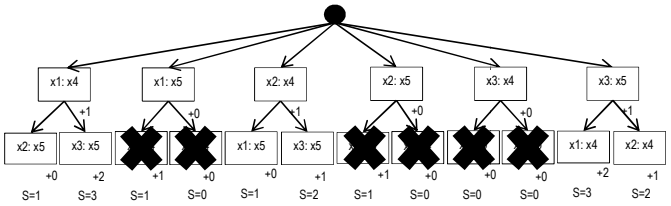


Figure 8 Backtracking with TypePrunning heuristic

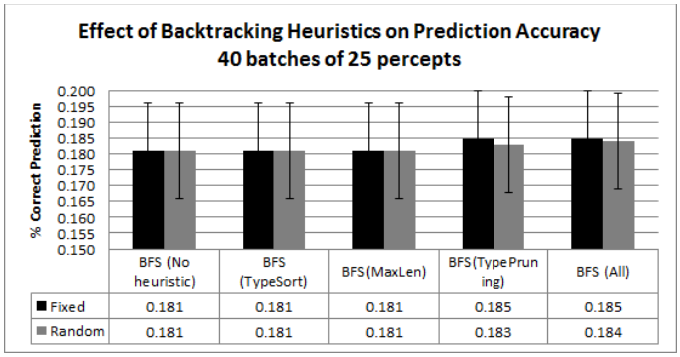


Figure 9 Effect of heuristics on prediction accuracy

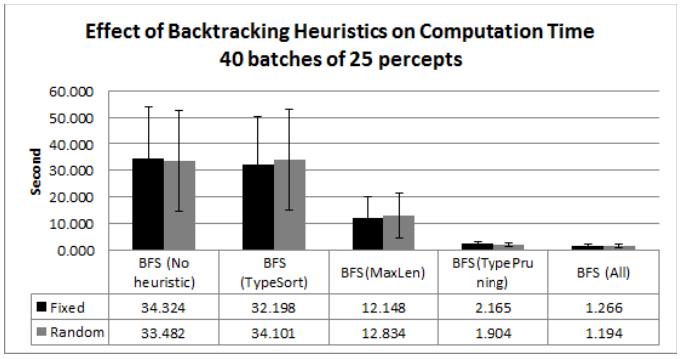


Figure 10 Effect of heuristics on computation time

We explored a best first search (BFS) method to attempt to improve the substitution method to cross space mapping. The best first search is similar to the backtracking, but with a priority queue for the fringe. The fringe is sorted based on the similarity score. Figure 11 illustrates the state of the fringe after processing the first level of the tree. As in backtracking, we compared the effect of the three heuristics on the prediction accuracy and computation time. Since the fringe is sorted, nodes at the end of the queue have lower similarity scores. This raises the question of whether we can ignore some of the nodes, and if so, how many. We ran some experiments with maximum fringe size of one, 1,000, 2,000 and 10,000. We know that the maximum fringe size is less than 8,000 in our task of 100 percepts. It turns out that there is no significant difference on the prediction accuracy for all four fringe size. However, the computation time is drastically different between fringe size one and fringe size 10,000. The results are described in Figure 12 and Figure 13. The computation time for BFS with fringe size 10000 is 10 times more than backtrack due to the sorting requirement. We use insertion sort. However, the computation time for fringe size one is much lower than all backtracking configurations, and yet able to achieve similar prediction accuracy. In our task, the BFS with fringe size one that includes all three heuristics appears to be the best method for the substitution method for cross space mapping. It turns out that, a fringe size of two is able to achieve exactly the same accuracy as fringe size 10,000.

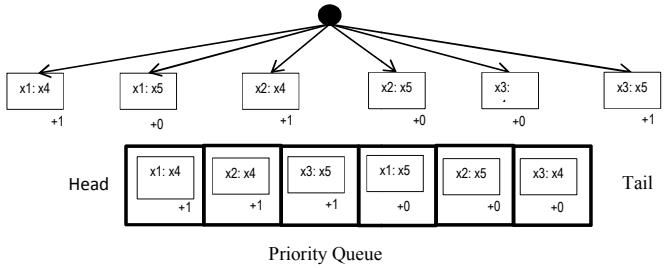


Figure 11: State of the priority queue after processing the first level of the tree

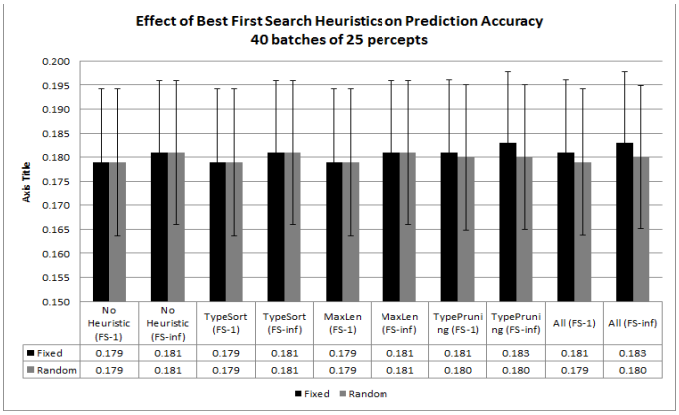


Figure 12: Effect of fringe size and heuristics on prediction accuracy for best first search. FS-1 means fringe size = 1. FS-inf means fringe is infinity.

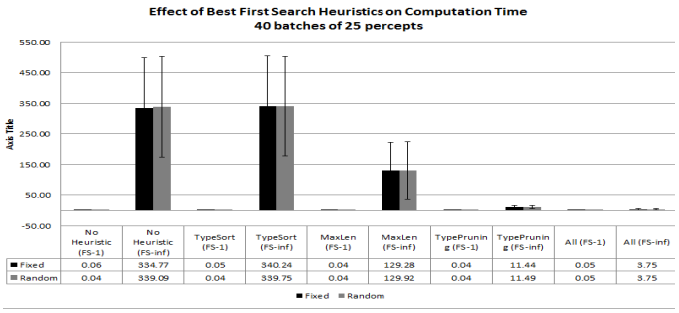


Figure 13: Effect of fringe size and heuristics on computation time for BFS. FS-1(fringe size = 1). FS-Inf(fringe is infinity)

We also tried an ASTAR method. In Figure 7, we see that the potential binding (x1:x4) and (x2:x4) have the same score. In the priority queue, it is possible that (x2:x4) is before

(x1:x4). Hence, we use a simple heuristic for (x1:x4) to appear before (x2:x4) after sorting. In BFS, the fringe is sorted based on the similarity score, known as the g-score. In ASTAR, the fringe is sorted based on f-score = g-score + h-score. The h-score in this task is computed as follow: in each node in the current path, every node that has a type mismatch will contribute a negative one (-1) score to the f-score. In Figure 7, (x1:x4) does not have type mismatch while (x2:x4) has type mismatch. F-score(x1:x4) = 1 – 0 = 1. F-score(x2:x4) = 1 – 1 = 0. The heuristic for ASTAR is admissible because it is either optimistic or zero. When there is a type mismatch, there will be more atoms difference than just the type atom. The sanity check can be illustrated with an example. Suppose we have two paths A [(x1:x5)] and B [(x1:x5),(x3:x4)] where B is the consequence of adding one node to A. These paths will lead to zero similarity score. Path A will have a f-score of 0 – 1 = -1 while path B will have a f-score of 0 – 2 = -2. A path that accumulates more mismatched nodes will have a lower score. In our experiments with A* and the heuristics, there is no effect on the prediction accuracy. There is a significant improvement in the computation time. The results of a paired t-test, comparing the computation time of BFS and A* are shown in Figure 14.

| | *BFS(MaxLen) FS=10000 Fixed* | *ASTAR(MaxLen) FS=10000 Fixed* |
|---|---|---|
| Mean | 129.2776 | 44.71395 |
| Variance | 346090.7684 | 37821.28 |
| Observations | 40 | 40 |
| df | 39 | 39 |
| F | 9.150688935 | |
| P(F<=f) one-tail | 1.341E-10 | |
| F Critical one-tail | 1.704465067 | |

Figure 14: Paired T-test comparing BFS and A* using heuristics MaxLen and fringe size of 10,000.

We implemented one final approach for cross space mappings: Structural Mapping (SM). In SM, all constants are replaced by their types. An atom maps to another atom if they have identical relation and there are corresponding constants from both atoms at their respective position that are of the same type. Such a mapping constitutes one common atom. Figure 16 illustrates the structures for both concepts in Figure 15.

### C. *Generic space*

The generic space contains the common atoms in both concepts. We look for a situation that maximizes the generic space. It is the most similar situation. In the case of multiple situations that share the same similarity score, the earliest one is used.

### D. *The Blend*

Once a previous situation is selected, the frame is used in the blend. The constants of the frame come from the current situation. Hence, constants from the selected previous situation are substituted by constants from the current situation. The constant substitution mappings are determined by the cross space mapping process. In the case of structural mapping, one constant will be substituted by another if both are of the same type. The predictive target atom is generated in the process since it is part of the previous situation. An example is given in Figure 17.

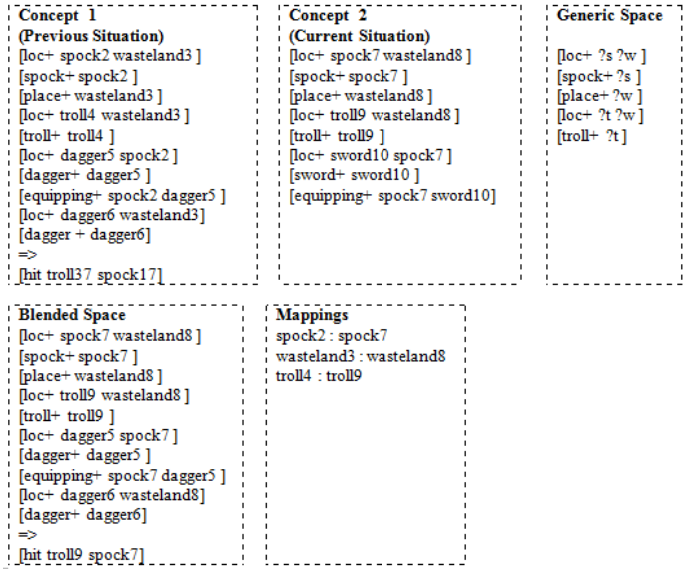


Figure 15 Concept 1 represents a situation that has a structure that best match the current situation in Concept 2

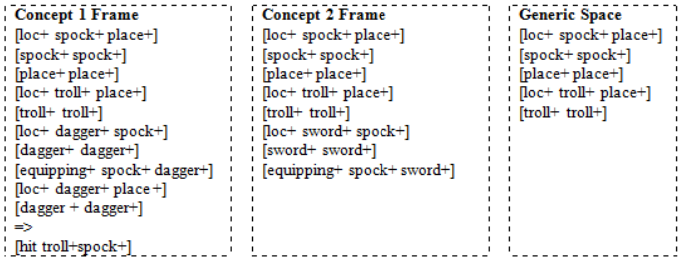


Figure 16 Structures of concept 1, concept 2 and generic space generated by structural mapping

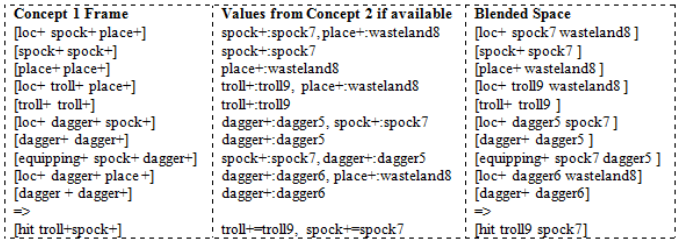


Figure 17: Generating the blend by using the frame from concept 1, and constant from concept 2. Constants mapping are identified based on constant type.

### E. *Back-projection*

The back-projection adds counterfactual details into the current situation, which in our case, is the predictive target atom.

### F. *Results*

In this experiment, we use 40 batches of 100 percepts, in order to compare with the earlier prediction techniques. The results of four single scope blending predictors (Backtrack (SSB-BT), BFS (SSB-BFS), ASTAR (SSB-ASTAR) and structural mapping (SSB-SM)) are described in Figure 18, alongside with the other predictors described earlier. The SSB-BT, SSB-BFS and SSB-ASTAR technique achieve the best prediction accuracy of 28%. The SSB-BFS and SSB-ASTAR

are the most robust techniques here that achieve good prediction accuracy at a reasonable time. The SSB-SM also seems to be a good approach. Although it's accuracy is 4% lower, the time to compute is better than the SSB-BFS and SSB-ASTAR. The single scope blending approaches achieve better performance because they make good use of the structural properties during the situation selection and inferencing processes. Unlike previous techniques that either require exact matching, or treat each atom as one proposition, single scope blending network approaches identify the previous situation that has the most similar structure, and uses the structure to make a prediction.

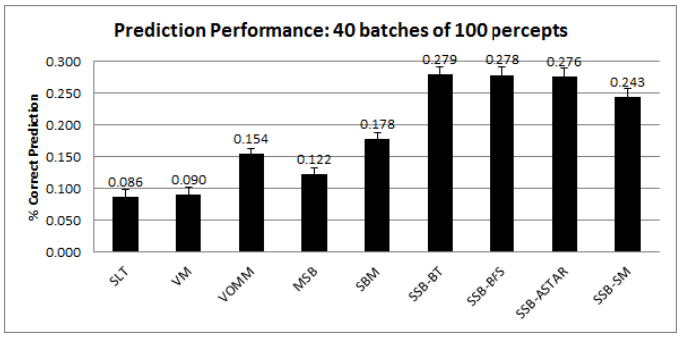


Figure 18 Comparison of Prediction Accuracy. SSB = Single Scope Blending, SSB-Sub: Substitution, SSB-SM: Structural Mapping, SSB-BFS-1: Best First Search with Fringe Size = 1, SSB-BFS-1000 with Fringe Size =1000.

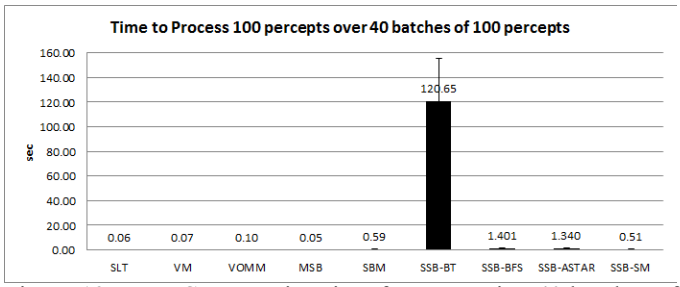


Figure 19 Mean Computation time for processing 40 batches of 100 percepts.

## VI. Conclusions & Future Work

We have presented a situation learning approach to learning relational time series by decomposing them into a set of situations. This approach can potentially handle the challenging characteristics of relational time series. With the set of situations in the agent memory, the inferencing process reduces to that of situation matching. It is only after the situation learning is employed that we can even possibly use established inferencing techniques such as Markov Chains and Bayesian Inferencing. However, even with situation learning, many current approaches have low predictive power because of their inability to predict new atoms for novel situation. Therefore, we developed analogy approaches to prediction that leverage the relational properties of the atomic structure. Finally, we have shown that the analogical approaches outperformed the other approaches significantly.

For future work, we have recently been introduced to the Event Segmentation Theory (EST) [8], which describes how human cognitively learn from the sequence of inputs. It will be interesting to develop a computational model of the EST and compare it with the situation learning approach. In addition, Conceptual Blending's double scope network can potentially contribute to more novel predictions. This may help us to address the issue of multiple situations that share the same similarity score.

## VII. Acknowledgement

This research effort is partially supported by the Temasek Defence Systems Institute, a strategic alliance between National University of Singapore and US Naval Postgraduate School. http://www.tdsi.nus.edu.sg/

## References

[1] P. Buhlmann, and A. J. Wyner, "Variable Length Markov Chains", The Annals of Statistics, Vol. 27, No. 2, 480-513, 1999.

[2] C. Darken, "Towards Learned Anticipation in Complex Stochastic Environments", Proceedings of AIIDE 2005.

[3] G. Fauconnier, and M. Turner, "The Way We Think: Conceptual Blending and The Mind's Hidden Complexities", ISBN-10: 046508785X, New York: Basic Books, 2004.

[4] L. Getoor, and B. Taskar, "Introduction to Statistical Relational Learning", MIT Press, ISBN-13: 978-0262072885, 2007.

[5] H. Jaeger, M. Zhao, and A. Kolling, "Efficient estimation of OOMs", In Proceedings of NIPS, 2005.

[6] H. Khosravi, and B. Bina, "A Survey on Statistical Relational Learning", In Proceedings of Canadian Conference on AI'2010. pp.256~268, 2010.

[7] I. Spanczer, "Observable Operator Models", Austrian Journal Of Statistics Volume 36, pp41–52, 2007.

[8] J. M. Zacks, and J. Q. Sargent, "Event perception: A theory and its application to clinical neuroscience", Psychology of learning and motivation (Vol. 53), 2009.

[9] J. R. Ullmann, "An algorithm for subgraph isomorphism", Journal of the ACM 23 (1): 31–42, 1976