ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-429

Spatio-Temporal Stream Reasoning with Incomplete Spatial Information

Fredrik Heintz and **Daniel de Leng**
IDA, Linköping University, Sweden

Abstract. Reasoning about time and space is essential for many applications, especially for robots and other autonomous systems that act in the real world and need to reason about it. In this paper we present a pragmatic approach to spatio-temporal stream reasoning integrated in the Robot Operating System through the DyKnow framework. The temporal reasoning is done in Metric Temporal Logic and the spatial reasoning in the Region Connection Calculus RCC-8. Progression is used to evaluate spatio-temporal formulas over incrementally available streams of states. To handle incomplete information the underlying first-order logic is extended to a three-valued logic. When incomplete spatial information is received, the algebraic closure of the known information is computed. Since the algebraic closure might have to be re-computed every time step, we separate the spatial variables into static and dynamic variables and reuse the algebraic closure of the static variables, which reduces the time to compute the full algebraic closure. The end result is an efficient and useful approach to spatio-temporal reasoning over streaming information with incomplete spatial information. [1]

1 Introduction

Spatial and temporal reasoning are central and well-studied topics in AI. The focus is usually on how to represent spatial, temporal and spatio-temporal information and how to efficiently reason with the information in a given knowledge base. In this paper we address the problem of *qualitative spatio-temporal stream reasoning*, i.e., incremental spatio-temporal reasoning over streams of information.

Our main application domain is autonomous unmanned aircraft systems (UAS). Both the information available to and the reasoning required for these autonomous systems are fundamentally incremental in nature. A flow of incrementally available information is called a *stream* of information. To draw relevant conclusions and react to new situations with minimal delays incremental reasoning over streams is necessary. We call such reasoning *stream reasoning*.

Reasoning about time and space is essential for autonomous systems acting in the real world. Consider for example monitoring the constraint that a UAS is not allowed to fly at a low altitude over urban areas for more than 3 minutes. This could be reformulated as: If the area occupied by the UAS overlaps an urban area then within 3 minutes the UAS should either be at a high altitude or the area it occupies should be disconnected from the urban area.

There exist many spatial and temporal formalisms. This work focuses on integrating qualitative spatial reasoning using the Region Connection Calculus RCC-8 [16] with the Metric Temporal Logic (MTL) [14]. RCC-8 captures topological reasoning over two dimensional convex regions and is probably the most used spatial reasoning formalism. MTL is selected since it supports temporal quantification over intervals of time. Another reason is that we already have a working solution to temporal stream reasoning using MTL [4].

The general idea of our approach to spatio-temporal stream reasoning is to perform the spatial reasoning within a time-point first and then perform temporal reasoning over the result. This makes it possible to evaluate MTL formulas containing RCC-8 relations where qualitative spatial reasoning is required to deduce implicit RCC-8 relations from the available incomplete spatial information. This clearly separates the spatial and the temporal reasoning which greatly simplifies changing the underlying spatial and temporal formalisms. This closely resembles the spatio-temporal language ST_0 [18].

To support spatio-temporal stream reasoning with incomplete spatial information we need to solve two problems. First, we need to extend traditional qualitative spatial reasoning over a fixed knowledge base to qualitative spatial stream reasoning. This is required to infer implicit information from the available incomplete information. Second, we need to extend our progression-based temporal stream reasoning to handle states containing disjunctive information. This is necessary since even after qualitative spatial reasoning the spatial information might be incomplete.

The main contribution of this work is a pragmatic approach to spatio-temporal stream reasoning integrated with DyKnow [7, 10, 12] and the Robot Operating System (ROS) [15]. The approach provides solutions both to the problem of qualitative spatial stream reasoning and to progression of metric temporal logical formulas over states with disjunctive information.

The rest of the paper is structured as follows. First, we provide a brief background to qualitative spatio-temporal reasoning, DyKnow and our existing approach to temporal stream reasoning. Then, we present a solution to the problem of qualitative spatial stream reasoning. After that we present our solution to the problem of temporal stream reasoning with disjunctive information. These solutions are then combined into an approach for spatio-temporal stream reasoning. Before concluding we present some empirical results regarding qualitative spatial stream reasoning.

2 Qualitative Spatio-Temporal Reasoning

Qualitative spatio-temporal reasoning is concerned with reasoning over time and space, in particular reasoning about spatial change [3].

[1] This work is partially supported by grants from the National Graduate School in Computer Science, Sweden (CUGS), the Swedish Aeronautics Research Council (NFFP6), the Swedish Foundation for Strategic Research (SSF) project CUAS, the Swedish Research Council (VR) Linnaeus Center CADICS, ELLIIT Excellence Center at Linköping-Lund for Information Technology, and the Center for Industrial Information Technology CENIIT.

Several qualitative spatio-temporal reasoning formalisms have been created by combining a spatial formalism with a temporal. Examples are STCC [6] and ARCC-8 [2] which both combine RCC-8 with Allen's Interval Algebra [1]. A qualitative representation provides a more abstract representation which reduces the complexity of the reasoning by focusing on the salient aspects. It also handles some forms of uncertainty by considering equivalence classes rather than values, and it provides a natural human-computer interface as people often think and communicate in terms of qualitative representations.

In this work we are interested in changes in topological relations between spatial regions. The Region Connection Calculus RCC-8 is the most well known approach to this type of spatial reasoning [16]. RCC-8 reasons about the relation between regions that are non-empty regular, closed subsets of a topological space, and can consist of more than one piece. RCC-8 has eight base relations (see Figure 1) which are jointly exhaustive and pairwise disjoint: DC (DisConnected), EC (Externally Connected), PO (Partial Overlap), EQ (EQual), TPP (Tangential Proper Part), NTPP (Non-Tangential Proper Part), and their inverse relations TPPi and NTPPi. The set of RCC-8 relations corresponds to all possible subsets of the base relations, where each subset is interpreted as the union of its relations.

The two main approaches to temporal reasoning is Allen's Interval Algebra and temporal modal logics. The modal logics usually extend the underlying propositional or first-order logic with temporal operators such as $\bigcirc$ ("next"), $\Diamond$ ("eventually"), $\Box$ ("always") and U ("until"). Metric Temporal Logic [14] extends first order logic with temporal operators that allows metric temporal relationships to be expressed. For example, F should hold within 30 seconds $\Diamond_{[0,30]}\,F$ and F' should hold in every state between 10 and 20 seconds from now $\Box_{[10,20]}\,F'$. Informally, $\Diamond_{[\tau_1,\tau_2]}\,\phi$ holds at τ iff ϕ holds at some $\tau' \in [\tau+\tau_1, \tau+\tau_2]$, while $\Box_{[\tau_1,\tau_2]}\,\phi$ holds at τ iff ϕ holds at all $\tau' \in [\tau+\tau_1, \tau+\tau_2]$. Finally, $\phi\,\mathsf{U}_{[\tau_1,\tau_2]}\,\psi$ holds at τ iff ψ holds at some $\tau' \in [\tau+\tau_1, \tau+\tau_2]$ such that ϕ holds in all states in (τ, τ').

The spatio-temporal formalism that is most relevant for this paper is the ST_i family of spatio-temporal languages initially proposed by Wolter and Zakharyaschev [18]. These languages combine RCC-8 with the propositional temporal logic PTL. In ST_0 RCC-8 relations can be temporally quantified. For example, $\Box\,PO(Sweden, Europe)$ states that it is always the case that Sweden is part of Europe. The expressive power of ST_0 is restricted to RCC-8 relations with region variables from the same time-point. To support spatial relations between region variables from different time-points ST_1 is introduced. It allows applications of the next-time operator $\bigcirc$ not only to formulas but also to region variables. Thus, arguments of the RCC-8 predicates are now region terms, which consist of a region variable that may be prefixed by an arbitrarily long sequence of $\bigcirc$ operators. Using ST_1 it is possible to state that a region X never changes $\Box\,EQ(X, \bigcirc X)$. The final member of the ST_i family is ST_2 where region variables may be prefixed by either $\Box$, $\Diamond$ or $\bigcirc$. The meaning of a region term $\Diamond\,r$ is the region corresponding to the union of every instance of r and $\Box\,r$ is then the intersection of every instance of r. The ST_i family can further be extended to ST_i^+ by allowing boolean combinations of region terms. This allows formulas such as $EQ(Scandinavia, Denmark \vee Finland \vee Iceland \vee Norway \vee Sweden)$. The ST_i family of languages can also be expressed as a multi-modal logic [2].

3 DyKnow and Temporal Stream Reasoning

DyKnow helps organize the many levels of information and knowledge processing in a distributed robotic system as a coherent network of processes connected by streams [7, 8, 10, 12]. The streams contain time-stamped information and may be viewed as representations of time-series data. Computational units combine streams by applying functions, synchronization, filtering, aggregation and approximation. The processing is done at many levels of abstraction, often beginning with low level quantitative sensor data and resulting in qualitative data structures which are grounded in the world and can be interpreted as knowledge by the system.

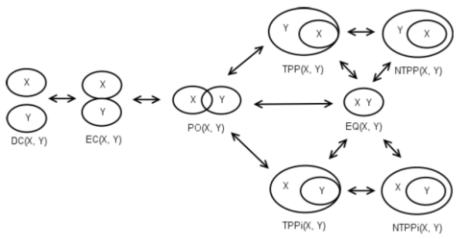

Figure 1. The RCC8 relations and their continuity network.

DyKnow uses progression of metric temporal logic formulas for incremental temporal reasoning over streams [4]. This provides real-time incremental evaluation of logical formulas as new information becomes available. The semantics of these formulas are defined over infinite state sequences. Formulas are therefore incrementally evaluated using progression over a stream of timed states. The result of progressing a formula through the first state in a stream is a new formula that holds in the remainder of the state stream if and only if the original formula holds in the complete state stream. If progression returns true (false), the entire formula must be true (false), regardless of future states. Even though the size of a progressed formula may grow exponentially in the worst case, it is always possible to use bounded intervals to limit the growth. It is also possible to rewrite formulas which limits the growth for common formulas [7].

A temporal logic formula consists of symbols representing variables, sorts, objects, features, and predicates besides the symbols which are part of the logic. Features may for example represent properties of objects and relations between objects. Consider $\forall u \in \mathsf{UAS} : u \neq \mathsf{uas1} \rightarrow \Box\,\mathsf{XYDist}[u, \mathsf{uas1}] > 10$, which has the intended meaning that all UASs, except uas1, should always be more than 10 meters away from uas1. This formula contains the variable u, the sort UAS, the object uas1, the feature XYDist, the predicates $\neq$ and $>$, and the constant value 10, besides the logical symbols. To evaluate such a formula an interpretation of its symbols must be given. Normally, their meanings are predefined. However, in the case of stream reasoning the meaning of features can not be predefined since information about them becomes incrementally available. Instead their meaning has to be determined at run-time. To evaluate the truth value of a formula it is therefore necessary to map feature symbols to streams, synchronize these streams and extract a state sequence where each state assigns a value to each feature [7].

DyKnow also supports automatically mapping features in a formula to streams in a system based on their semantics, which we call *semantic grounding* [8, 9, 11]. By introducing semantic mapping between ontologies from different UASs and reasoning over multiple related ontologies it is even possible to find relevant streams distributed among multiple UASs [11].

4 Qualitative Spatial Stream Reasoning

In this section we address the problem of qualitative spatial reasoning over streams of spatial information. The spatial information can vary over time and may be incomplete. In this paper we assume that the available spatial information is qualitative. We have started to consider the case of quantitative spatial information, usually some form of geometric information, and mixed qualitative and quantitative spatial information, but this is left for future work.

The general idea is to collect all the relevant spatial relations about a particular time-point, use a qualitative spatial reasoner to compute the algebraic closure of the set of known relations to check the consistency and infer implicit relations, and then generate streams based on the result of this spatial reasoning. The algebraic closure rather than complete backtracking is chosen to reduce the computational overhead and make the approach practical. It is straightforward to use a complete approach instead if desirable.

To compute the algebraic closure can be time-consuming if the number of variables is large. To increase the efficiency we divide the variables into two sets, static and dynamic variables. A *static variable* represents a spatial region that does not change, which means that the relations between the static variables are fixed. A *dynamic variable* is one whose corresponding region changes over time. It could for example be the variable corresponding to the area that is in view of a particular sensor on a particular platform. The algebraic closure of the static variables is only computed once. Each time the dynamic relations changes, the algebraic closure of the complete set of variables is recomputed starting from the algebraic closure of the static variables.

We provide our solution both as a collection of ROS services and as a computational unit integrated with DyKnow. The reasoner that we use is GQR [5] which is a fast general qualitative constraint reasoner that supports RCC-5, RCC-8, and RCC-23 among other qualitative representations. The computational unit takes as input a stream of sets of spatial relations and as output produces a stream of spatial relations corresponding to the algebraic closure of the current set of relations. This extends DyKnow with support for spatial reasoning over streams of spatial information in any of the qualitative spatial representations supported by GQR.

4.1 A Qualitative Spatial Reasoning ROS Service

The proposed ROS-based qualitative spatial reasoning service provides a set of service calls for creating, updating and querying qualitative spatial knowledge bases. The service keeps track of the evolution of the knowledge bases to support time-varying spatial information. Since GQR is used to compute the algebraic closure it can handle any qualitative calculus supported by GQR.

ROS Services

```
create_kb(string kbname, string algebra,
          Relation[] static_constraints,
          int32 static_variable_threshold_ms)
 : ExitStatus exit_status

set_static_variable_threshold_ms(int32 new_threshold)
 : ExitStatus exit_status,
   int32 old_threshold_ms

replace_constraints(string kbname, time t,
                    Relation[] constraints)
 : ExitStatus exit_status

remove_variable(string kbname, string variable)
 : ExitStatus exit_status

compute_algebraic_closure(string kbname, time t)
 : ExitStatus exit_status,
   bool result

get_relations_at(string kbname, time t,
                 string[] variables)
 : ExitStatus exit_status,
   Relation[] relations

get_current_relations(string kbname,
                      string[] variables)
 : ExitStatus exit_status,
   Relation[] relations
```

The service call create_kb creates a new named knowledge base for a particular qualitative algebra. Supported algebras include RCC-5, RCC-8, RCC-23, point algebra, and OPRA [5]. It is also possible to add a set of static constraints to the knowledge base and to define the threshold for when to assume that a variable is to be considered static. The threshold can be changed with the service call set_static_variable_threshold_ms. It is important to notice that this only affects the performance of the qualitative reasoning. It is always possible to change the relation between a pair of variables, but in the worst case the full algebraic closure has to be computed.

The service call replace_constraints replaces the constraints for a set of variables. To remove the constraint between a pair of variables, add a constraint containing all the base relations. If a constraint contains a variable not previously in the knowledge base it will be added. To remove a variable use remove_variable.

compute_algebraic_closure computes the algebraic closure of a knowledge base at a particular time-point. For most applications it is not necessary to call this explicitly since it is called when needed.

The service calls get_relations_at and get_current_relations return all the relations for the given variables either for a specific time-point or for the current content of the knowledge base. If the set of variables is empty the set of all relations for all variables is returned. If needed, the algebraic closure will be computed.

4.2 Dynamically Updating Spatial Knowledge Bases

To update a spatial knowledge base two things are needed: (1) the new information must be stored efficiently and (2) all implicit relations must be inferred. Since information might be available asynchronously we need to store information related to multiple time-points. This also makes it possible to query past spatial relations.

An in-memory database of spatial relations between regions (variables) over time is created to store the complete history of spatial relations efficiently. For each pair of region variables v_1 and v_2 such that $v_1 \preceq v_2$ create an ordered array of pairs $\langle t, s_b \rangle$, where t is the time-point when the set of relations changed and s_b is the new set of relations represented as a bit vector. The $\preceq$ ordering on the variables could for example be lexicographical ordering. This allows finding a particular relation in $O(\log |v|)$ time. If there are at most 64 base relations then the bit vector can be represented by an unsigned integer on modern machines. This requires $O(|v|^2 k)$ space, where k is the maximum number of changes in the spatial relations. Since the updates to the spatial relations are ordered in time, the array can be updated in constant time for each relation. The relations between a pair of variables at a particular time t can be found using binary search over the array for the pair of variables and requires $O(\log k)$ time. The current set of relations between a pair of variables can be found in $O(1)$ time since it is always the last element in the array.

An alternative is to replace the arrays by circular buffers that store a fixed number of changes to limit the amount of memory used.

To efficiently infer implicit relations we use the facts that the algebraic closure for the same set of variables must be computed many times (every time some of the variables have changed) and relations between variables that have not changed are the same. If the set of variables is partitioned into those that are static and those that are dynamic, it is enough to compute the algebraic closure of the constraints involving only static variables once and then add the constraints involving at least one dynamic variable when they have changed and compute the new algebraic closure. The effect is that there is an initial cost of computing the static part while the cost for each update is reduced. In the empirical evaluation section we quantify these gains.

To simplify the usage of the spatial reasoning service while maximizing the performance, the knowledge base maintains the set of variables V_s whose constraints have not been changed for at least s milliseconds. The appropriate threshold depends on the application, including how fast things change and the size of the constraint problem. The threshold s can be configured at run-time using the set_static_variable_threshold_ms service.

Formally, let V^t be the set of all variables at time t and C^t be the set of all (binary) constraints on these variables at time t. The set V^t is partitioned into V_s^t and V_d^t, where V_s^t is the set of static variables and V_d^t is the set of dynamic variables at time t. C^t is partitioned into C_s^t and C_d^t, where C_s^t is the set of constraints where both variables belong to the set V_s^t and C_d^t is the set of constraints where at least one variable belong to the set V_d^t. Further, let AC_s^t denote the algebraic closure of the variables V_s^t and the constraints C_s^t and AC^t denote the algebraic closure of the variables V^t and the constraints C^t. Then, AC^t can be computed from AC_s^t by adding the constraints C_d^t and computing the algebraic closure. As we show in the empirical evaluation section, this improves the efficiency since $AC_s^{t+1} = AC_s^t$ as long as V_s^t does not change.

If a variable v is added to V_s at $t+1$, then C_s^{t+1} is C_s^t union the set of constraints involving v and another variable in V_s^t. AC_s^{t+1} can then efficiently be compute from AC_s^t. If a variable v is removed from V_s^t then the algebraic closure for the new set V_s^{t+1} is computed.

5 Temporal Reasoning over Streams of Disjunctive Information

The metric temporal reasoning we use does not explicitly handle incomplete information such as disjunctive information. The semantics is defined over states where each state assigns each ground term exactly one value. We have extended the temporal reasoning to handle states where a ground term might have multiple values. To achieve this we extend the metric temporal logic from a standard two valued logic to a three valued Kleene logic [13]. The three values are true, false and unknown, where unknown represents that the value is either true or false but it is not known which. The extension consists of three parts: (1) Extend the state representation from assigning each ground term to a value to assigning each ground term a set of values. (2) Extend the progression algorithm to handle the new state representation. (3) Extend the progression algorithm to handle the three valued logic.

The new extended state representation consists of a time-stamp and a collection of key-value pairs where each key is a ground term and each value is a collection of values. If the collection only contains one value then the value of that term is completely known. If the collection contains multiple values then the actual value of the ground term is one of those, but it is not known which. This allows incomplete information to be represented as a disjunction of possible values. This is a strict extension of the previous representation where a set with only one value is equivalent to assigning the term a particular value.

The grammar for our metric temporal logic is the following:
$\phi \;:=\; f \mid f_1\, rop\, f_2 \mid \neg\phi \mid \phi_1\, lop\, \phi_2 \mid \forall v.\phi \mid \phi_1 \,\mathsf{U}_{[a,b]}\, \phi_2 \mid top_{[a,b]}\phi$, where f is a ground feature term, $rop \in \{<, \leq, =, \geq, >\}$, $lop \in \{\wedge, \vee\}$, $top \in \{\Box, \Diamond\}$, and a and b are integers.

The progression algorithm uses the state to get the value of ground features (f in the grammar). The spatial relations are considered to be features and do not require any special treatment during progression. Previously each ground feature was given a particular value from the state. With the extended state representation it may now have a set of possible values. The progression algorithm handles this by extending its internal representation to support sets of values. The truth value of an atomic formula f is only defined for ground features whose value is a boolean. The extended progression algorithm evaluates this atomic formula to true iff the set of values for f only contains true, to false iff the set of values for f only contains false and otherwise to unknown. The truth value of a formula $f_1\, op_1\, f_2$ is true (false) iff op_1 is true (false) for every possible value of f_1 and f_2. If it is neither true nor false then it is unknown.

Finally, the progression algorithm is extended to evaluate the logical operators $\neg$, $\vee$ and $\wedge$ according to the Kleene semantics [13]. Since temporal operators are expanded through progression to conjunctions and disjunctions no further changes are needed.

With these extensions our progression algorithm for metric temporal logic is extended to use a three-valued first order Kleene logic instead of a standard first order logic. The major limitation is that atomic formulas are treated independently. This means that a formula $DC(a,b) \vee PO(a,b)$ would be evaluated to unknown even if the set of possible relations between the regions a and b are $\{DC, PO\}$ and it would be possible to deduce that the formula is true. We are currently working on a solution to this.

6 Spatio-Temporal Stream Reasoning with Incomplete Spatial Information

By combining qualitative spatial reasoning over streams and progression over disjunctive information we provide an approach to spatio-temporal stream reasoning with incomplete spatial information. The approach separates the spatial reasoning from the temporal reasoning and corresponds to ST_0. The output of the spatial stream reasoning is a stream of sets of spatial relations, where each set represents all the spatial information available at a particular time-point. This stream is then synchronized with other streams of non-spatial information into a stream of states containing all the information necessary to evaluate a spatio-temporal formula. If a formula only contains spatial relations and temporal operators, then the formula can be evaluated directly over the output from the spatial stream reasoning module. Since we extended the progression algorithm it is possible to handle the fact that spatial information might be incomplete.

As a concrete example, consider monitoring that a UAS is never allowed to fly over an urban area for more than 3 minutes. This could be captured in the following formula: $\forall r \in \mathsf{UrbanRegion}\,\Box(PO(\mathsf{UAS}, r) \rightarrow \Diamond[0, 180\text{s}]DC(\mathsf{UAS}, r))$. This is an example formula which does temporal reasoning over spatial information only. An alternative safety constraint could be that if a UAS flies over an urban area then within 60 seconds it should be at an altitude of more than 100 meters. This could be expressed as $\forall r \in \mathsf{UrbanRegion}\,\Box(PO(\mathsf{UAS}, r) \rightarrow \Diamond[0, 60\text{s}](DC(\mathsf{UAS}, r) \vee \mathsf{altitude}(\mathsf{UAS}) > 100\text{m}))$. This is an example formula which also includes non-spatial information, in this case the altitude of the UAS. DyKnow handles units of measurements directly in formulas and can automatically transform streams containing information in

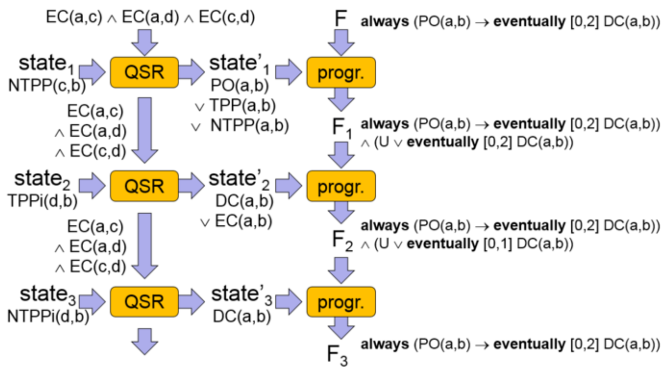
EC(a,c) ∧ EC(a,d) ∧ EC(c,d)
state1
NTPP(c,b)
QSR
state'1
PO(a,b)
∨ TPP(a,b)
∨ NTPP(a,b)
progr.
F always (PO(a,b) → eventually [0,2] DC(a,b))
EC(a,c)
∧ EC(a,d)
∧ EC(c,d)
F1 always (PO(a,b) → eventually [0,2] DC(a,b)) ∧ (U ∨ eventually [0,2] DC(a,b))
state2
TPPi(d,b)
QSR
state'2
DC(a,b)
∨ EC(a,b)
progr.
F2 always (PO(a,b) → eventually [0,2] DC(a,b)) ∧ (U ∨ eventually [0,1] DC(a,b))
EC(a,c)
∧ EC(a,d)
∧ EC(c,d)
state3
NTPPi(d,b)
QSR
state'3
DC(a,b)
progr.
F3 always (PO(a,b) → eventually [0,2] DC(a,b))

Figure 2. A qualitative spatio-temporal stream reasoning example.

one unit of measurement to another unit of measurement [9].

Figure 2 shows a complete example of evaluating the spatio-temporal formula $\Box(PO(a,b) \rightarrow \Diamond DC(a,b))$ given the static region variables a, c, d and the static relations $EC(a,c)$, $EC(a,d)$, and $EC(c,d)$. The spatial information in the first state is the spatial relation $NTPP(c,b)$ which after spatial reasoning gives that $PO(a,b)$ can be either true or false, i.e., is unknown. In the next state the spatial relation $TPPi(d,b)$ is given from which the conclusion that $DC(a,b)$ is unknown can be drawn. Finally, in the third state where $NTPPi(d,b)$ is given spatial reasoning concludes that $DC(a,b)$ is true. This example shows both the benefit of spatial reasoning as no explicit information about the relation between a and b is given and the use of three-valued progression over disjunctive information.

Since all the spatial information is collected and reasoned over in the spatial reasoning module it is only done once. The single stream of spatial information can be combined in many different ways with other streams to support the evaluation of many different spatio-temporal formulas.

7 Empirical Evaluation

To show the scalability of the spatio-temporal stream reasoning approach we have conducted a number of experiments. The purpose of the evaluation is to quantify the gain of partitioning the variables into static and dynamic. The performance of the temporal reasoning is only marginally changed by the extensions to the progression algorithm. The performance is mainly dependent on the number of nested temporal operators. For a formula with three nested temporal operators it takes on average about 60 milliseconds to progress 1000 instances of the formula over a state on the computer onboard our UAS. Due to space limitations we therefore focus on the spatial reasoning part and refer interested readers to [7, 8].

The performance of the spatial reasoning is mainly dependent on the number of variables, the average number of constraints (degree) and the average label size [17]. Using basically the same method as Renz and Nebel [17] we evaluate the effect of precomputing the algebraic closure of the static variables, compared to computing the whole algebraic closure for each time-step.

In the experiments we try to estimate the function $A(v, E(deg), E(l), r)$ by measuring the execution time on instances with the number of variables v, the expected degree $E(deg)$, the expected label size $E(l)$ and the ratio of dynamic variables r. The number of variables can be divided in a dynamic part $v_d = r \times v$ and a static part $v_s = v - v_d$. The expected degree is the expected number of constraints from a given dynamic variable to other variables. The expected label size is the expected size of the disjunction of RCC-8 relations for a given constraint between a dynamic variable and some other variable. In this evaluation we use $E(l) = 4.0$.

Because the static component only has to be computed once, we compare the case where all variables are dynamic to the case where there is a separation between static and dynamic variables, ignoring the time it takes to compute this static component. The mean performance results of the former are denoted by $A(v, E(deg), 4.0, 1.00)$. For the mean performance results of the dynamic component of the latter, the notation $A_d(v, E(deg), 4.0, r)$ is used. The performance experiments used values of $E(deg)$ ranging from 1 to 20 with step size 1 and values of v ranging from 20 to 500 with step size 20. The value of r was chosen to be constant, $r = 0.25$. For every combination, we took the population mean CPU time over 100 runs. The population mean was chosen to account for the difference in distribution between the satisfiable and unsatisfiable problem instances.

The evaluation compares the case of all variables being dynamic to the case when some are static. A selection of the evaluation results are shown in Figure 3.

The top-left graph in Figure 3 shows the performance of $A(v, E(deg), 4.0, 1.00)$ in CPU time. The graph shows a ridge at $E(deg) = 9$. This is where the phase-transition occurs, where the majority of instances flip from being satisfiable to being unsatisfiable.

In comparison, the top-right graph shows the performance of $A_d(v, E(deg), 4.0, 0.25)$ in CPU time. Note that this only shows the time needed by the dynamic component. For low degrees, the time needed surpasses that of the exclusively dynamic case. A potential explanation for this behavior is that the combination of a low degree and high number of dynamic variables combined with the completely known static part (i.e. a degree of $v_s - 1$ and label size $l = 1$ for the static component) makes for computation-intensive problem instances. For all other values of v and $E(deg)$ the performance is significantly improved.

A comparison of the two top-row graphs is shown in the bottom row of Figure 3. On the left the absolute performance increase is shown, and on the right the relative performance increase. The absolute comparison shows a clear decrease in performance when comparing the exclusively dynamic case to the separated case when the degree is low and the number of variables is high. However, in all other cases there is a performance increase, especially around the phase-transition area. The general performance increase is roughly 50 milliseconds. The relative performance increase graph on the right shows an improvement of about 35% in the phase-transition area, and an improvement of close to 100% for a low number of variables.

The results in Figure 3 show that the separation of dynamic and static variables for $r = 0.25$ generally leads to better performance, except in the case of a low degree with a high number of variables. The performance increase is at its highest around the phase-transition region where the more difficult problem instances reside. The performance increase is expected to be higher for lower values of r and decrease as r approaches 1.

8 Conclusions and Future Work

We have presented a pragmatic approach to spatio-temporal stream reasoning which handles incomplete spatial information. The temporal reasoning is done using Metric Temporal Logic and the spatial reasoning using RCC-8. The approach first does the spatial reasoning and then the temporal reasoning, in line with previous approaches to spatio-temporal reasoning such as ST_0. By separating the spatial and temporal reasoning and using the fast and general

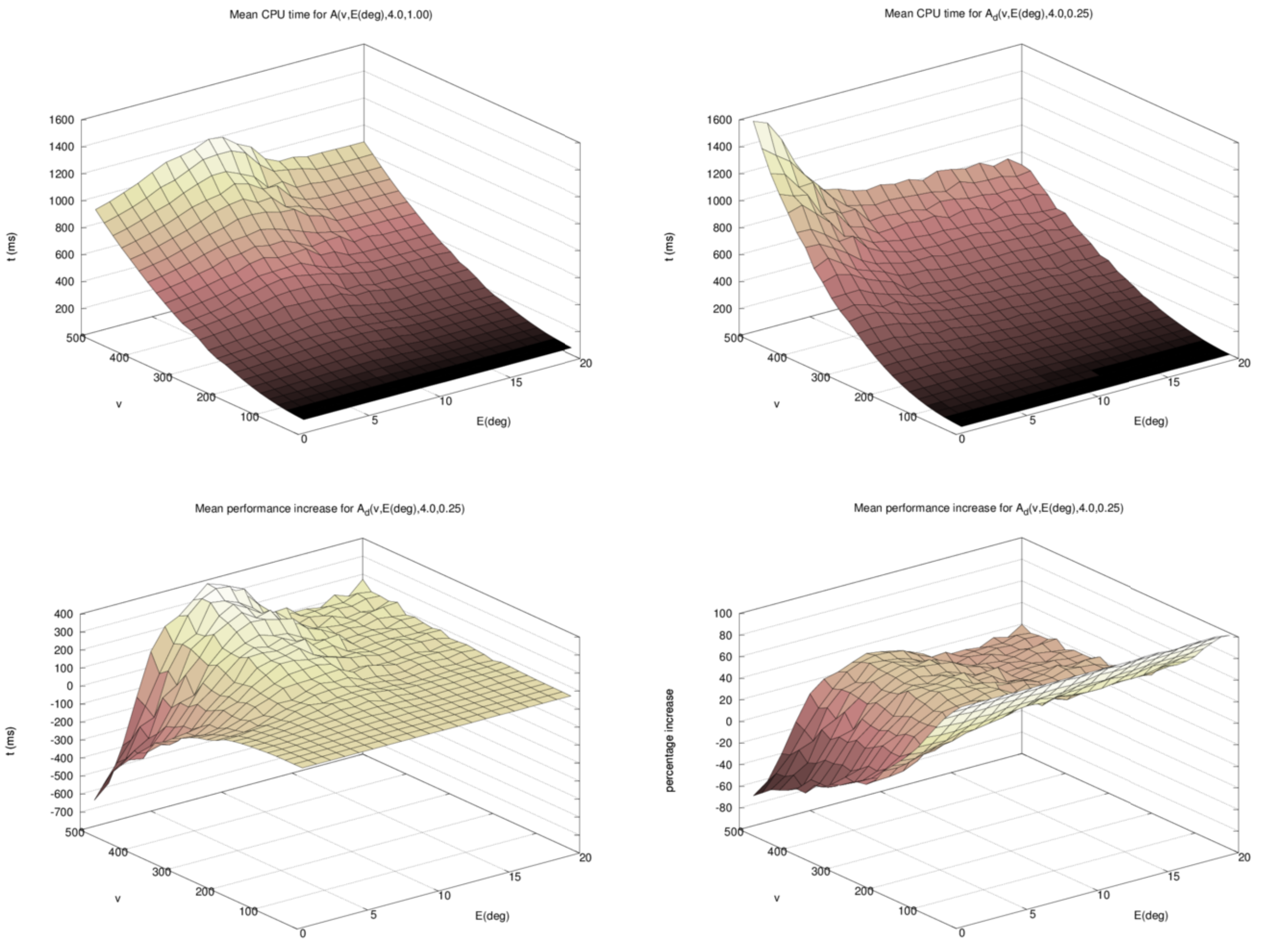

Figure 3. Comparison of performance when separating static and dynamic variables. The ratio of dynamic variables is fixed at $r = 0.25$. The top two graphs show the mean absolute CPU times and the bottom two graphs show the mean absolute and relative performance increase.

GQR constraint reasoner it is very easy to either use multiple different qualitative reasoning approaches together or replacing RCC-8 with another qualitative spatial approach. The approach has been integrated with the stream-based knowledge processing middleware DyKnow and the Robot Operating System.

This work opens up many interesting avenues for further research such as tighter integration of the spatial and temporal stream reasoning as well as even better support for handling incomplete information. In both cases, much can likely be gained by considering the whole formula or at least first order sub-formulas instead of individual atomic formulas. Another interesting direction is to combine quantitative and qualitative spatial reasoning. This would also open up for the possibility of supporting the rest of the ST_i family of spatio-temporal languages.

REFERENCES

[1] J. Allen, ‘Maintaining knowledge about temporal intervals’, *Commun. ACM*, **26**(11), 832–843, (1983).

[2] B. Bennett, A. Cohn, F. Wolter, and M. Zakharyaschev, ‘Multi-dimensional modal logic as a framework for spatio-temporal reasoning’, *Applied Intelligence*, **17**(3), 239–251, (2002).

[3] A. Cohn and J. Renz, ‘Qualitative spatial representation and reasoning’, in *Handbook of Knowledge Representation*, Elsevier, (2008).

[4] P. Doherty, J. Kvarnström, and F. Heintz, ‘A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems’, *J. of Auton. Agents and Multi-Agent Systems*, **19**(3), (2009).

[5] Z. Gantner, M. Westphal, and S. Wölfl, ‘GQR - a fast reasoner for binary qualitative constraint calculi’, in *Workshop Notes of the AAAI-08 Workshop on Spatial and Temporal Reasoning*, (2008).

[6] A. Gerevini and B. Nebel, ‘Qualitative spatio-temporal reasoning with RCC-8 and Allen’s interval calculus: Computational complexity’, in *Proc. ECAI*, (2002).

[7] F. Heintz, *DyKnow: A Stream-Based Knowledge Processing Middleware Framework*, Ph.D. dissertation, Linköpings universitet, 2009.

[8] F. Heintz, ‘Semantically grounded stream reasoning integrated with ROS’, in *Proc. IROS*, (2013).

[9] F. Heintz and D. de Leng, ‘Semantic information integration with transformations for stream reasoning’, in *Proc. Fusion*, (2013).

[10] F. Heintz and P. Doherty, ‘DyKnow: An approach to middleware for knowledge processing’, *J. of Intelligent and Fuzzy Syst.*, **15**(1), (2004).

[11] F. Heintz and Z. Dragisic, ‘Semantic information integration for stream reasoning’, in *Proc. Fusion*, (2012).

[12] F. Heintz, J. Kvarnström, and P. Doherty, ‘Bridging the sense-reasoning gap: DyKnow – stream-based middleware for knowledge processing’, *J. of Adv. Engineering Informatics*, **24**(1), (2010).

[13] S. Kleene, ‘On notation for ordinal numbers’, *Symbolic Logic*, (1938).

[14] R. Koymans, ‘Specifying real-time properties with metric temporal logic’, *Real-Time Systems*, **2**(4), 255–299, (1990).

[15] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, ‘ROS: an open-source robot operating system’, in *ICRA Workshop on Open Source Software*, (2009).

[16] D. Randell, Z. Cui, and A. Cohn, ‘A spatial logic based on regions and connection’, in *Proc. KR*, (1992).

[17] J. Renz and B. Nebel, ‘Efficient methods for qualitative spatial reasoning’, *J. of Artificial Intelligence Research*, **15**, 289–318, (2001).

[18] F. Wolter and M. Zakharyaschev, ‘Spatio-temporal representation and reasoning based on RCC-8’, in *Proc. KR*, (2000).